Performance improvement based on cross-organizational recommendations

Swinkels, G.J.A.

Award date:
2012

Department of Mathematics and Computer Science
Architecture of Information Systems Research Group

Performance Improvement based on Cross-Organizational Recommendations

Master's Thesis

ing. G.J.A. Swinkels

Supervisors:

dr. ir. B.F. van Dongen

ir. J.C.A.M. Buijs

dr. T.G.K. Calders

Eindhoven, March 2012

Abstract

Recommendations are used to improve the performance of an organization, based on its own historical data. Most recommendations make use of a transition system, which is a process model that is derived from data inside an event log. This research presents an approach that creates a collective transition system, a process model that contains data of multiple event logs. As a result, historical data of different organizations can be used for performance improvement.

Using configurable process models, control-flow patterns can be configured differently. In this way, similar organizations are obtained.

Software components are implemented to carry out experiments that investigate whether cross-organizational data of such similar organizations can also be used for performance improvement and to what extent the achieved performances differ. Plug-ins for the process mining tool ProM are developed that implement collective transition systems and a recommendation service called Multiple Log Recommender which provides recommendations to process instances of organizations. Executable process models are created that generate event logs and process executions which request and process recommendations of organizations.

Experiments with 5 artificial, similar organizations are executed that measure performance using various settings for historical data and percentage of recommendations to adhere to. The results show that performance gains are achieved when carrying out all recommendations of the Multiple Log Recommender, independent of whether these are derived from own or cross-organizational data. Moreover, configurations of control-flow patterns that allow for choices and parallel behaviour have a positive impact on the opportunity for performance improvement.

Keywords: recommendations, performance improvement, configurable process models, control-flow patterns, cross-organizational data

Preface

This master thesis is the result of my graduation project for the master's degree in Business Information Systems at Eindhoven University of Technology.

First of all, I would like to thank Joos Buijs for his guidance and availability during the entire project. His feedback was of great value. Second, many thanks go to my supervisor Boudewijn van Dongen for all his help and expertise. Third, I want to thank Toon Calders for being part of the examination committee. Also, thanks to the people of the Architecture of Information Systems research group who somehow contributed to this master thesis.

Next, I would like to thank all fellow students with which I collaborated throughout the master program.

Furthermore, big thanks go out to family and friends for all the good times during my spare time.

Last, but certainly not least, I want to thank my parents for always supporting me.

Guido Swinkels,
March 2012

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

This master thesis is the result of a graduation project in the Business Information Systems master program. It is performed at the Architecture of Information Systems (AIS) research group of the Mathematics and Computer Science department at Eindhoven University of Technology (TU/e). The AIS group conducts research on techniques, methods and tools for the design and analysis of systems that support business processes in organizations, also called process-aware information systems. A main focus is the area of process mining: a way to extract knowledge from systems by taking reality as the starting point (for instance by looking at recorded data of these systems) and generating process models from these. The master project has been carried out within this process mining context of the AIS group.

One of the many research directions of the AIS group is concerned with the architecture for processes that exchange data through the web. In light of this, the CoSeLoG project[1] was initiated. The main goal of this project is providing an environment that different municipalities can use for support of their business processes, while keeping own characteristics and settings. It attempts so by making use of configurable process models, which are described later on in this thesis, within a Software-as-a-Service environment. At the same time, such an environment offers a lot of interesting possibilities for the use of process mining techniques. For instance, research can be carried out which tries to discover if municipalities that carry out business processes in a similar way can learn from each other.

This work focuses on this enabled research direction and tries to investigate whether it is possible for organizations to make use of data of deviating processes (possibly from another organization or company) in order to improve their own business process execution.

1.1 Problem Description

A business process is a set of logically related tasks performed to achieve a defined business outcome [1]. Every organization has multiple business processes. Depending on the used scope, the number of business processes varies drastically. For instance, the processing of an insurance claim can be perceived as one process, but by using another scope, small parts of it can also be seen as a business process, for example the activities related to requesting information which

[1] For more information, see http://www.win.tue.nl/coselog/wiki/start

occur in the beginning of processing an insurance claim. A general rule of thumb specifies that organizations embed between 10 and 20 main processes.

There are several types of organizations, depending on the sector in which is operated and the collective goal that is pursuit. Organizations can be divided into different types, such as corporations, governments, non-governmental organizations and charities. Municipalities are examples of governmental organizations. Typically, municipalities all carry out similar business processes. These include the handling of requests for building permits, driving licenses, personal documents, birth registration etc. It would seem logical that these processes were executed in a similar fashion by every municipality. However, there are a number of factors that prevent this from actually being the case.

Divergent local policies and procedures are a reason for deviating executions of a business process. Other causes are due to differences in surface area, number and average characteristics of residents, size of the municipality, culture, demographics etc.

The factors given above are differences which cannot be changed or circumvented. However, there is an additional factor which leads to different business process executions that can be avoided: the separate choice of municipalities for a process-aware information system. Process-aware information systems support the execution of business processes and provide organizations insight in the current state of their business process executions. Currently, municipalities each make their own choice for such an information system.

The specific type of information system that is chosen has a direct effect on the way municipalities carry out the business processes supported by the system. Different underlying techniques and settings influence the level of suitability for cooperation with municipalities that use another process-aware information system. As a consequence, municipalities do not know how well they carry out their work and whether there is room for improvement of their business processes, since they cannot compare their process executions with those of other municipalities using a different system. Because municipalities are not competitors of each other and provide the same services to their residents, this is a missed opportunity. Intensive cooperation between municipalities about their approach can lead to improvement of their performance, because they can adapt their business process to better performing executions.

The CoSeLoG project tries to solve this problem by providing municipalities a shared environment. This shared environment still allows for local variations among the different business processes. At the same time, it offers a lot of possibilities for research, because the same techniques and settings are used, and data of process executions of different municipalities is available in the same format and can be compared with each other.

Yet, in order to make it possible for municipalities to learn from one another, more research is needed. However, unfortunately there is no existing work that provides information about whether it is possible that organizations which carry out a similar business process in a different way can make use of each other's process executions to improve their own business process executions.

1.2 Research Questions

In the previous section, it was discussed that currently there is no knowledge about whether it is possible to improve the performance of an organization by looking at data of other, similar organizations. This problem description leads us to the following research question:

> ***Research question:*** *Can organizations improve their performance by making use of historical data of other, similar organizations?*

This research question is divided into four smaller sub questions. By answering these sub questions, the research question itself is also answered.

First, an investigation needs to be carried out that addresses the question how it is possible to improve performance using recorded data. Therefore, sub question 1 is the following:

> ***Sub question 1:*** *How can historical data be used to improve performance?*

After solving this sub question, a next question that arises is related to how data of different business processes can be used together to base advices for improvement on. The resulting sub question reads as follows:

> ***Sub question 2:*** *How can historical data of different organizations be combined such that it can be used for performance improvements?*

A third research direction relates to the business processes of the organizations. Under which conditions are these processes actually similar enough, such that their historical data can be used for this work. Additionally, the effect of differences in control-flow configuration patterns of these processes on performance is researched. Sub question 3 therefore states:

> ***Sub question 3:*** *What are similar business processes and which differences in control-flow have an impact on the usability of the techniques for performance improvement?*

The final sub question, which is derived from the research question, concerns the use of recorded behaviour of organizations with varying control-flow configuration patterns. It questions which positive or negative effects on performance of an organization can occur when using historical data originating from other organizations.

> ***Sub question 4:*** *What is the difference between using historical data of the same or other, similar organizations?*

The following section describes the approach which is used in order to answer these research questions. Then, the outline of this document is presented in the final section of this chapter.

1.3 Research Approach

To answer the research questions, the following approach is taken:

Perform a literature study (Chapter 2)

First, existing techniques and literature are investigated in order to discover the current state of related topics in the research field.

Investigate an approach which improves performance based on historical data (Chapter 3)

An approach is investigated which is able to improve performance of a business process based on its own historical data. Furthermore, steps which need to be taken in order to use data of multiple organizations are discussed.

Identify notion of similar business processes (Chapter 4)

The next step is to research which variations of a business process can be derived. Possible configurations in terms of control-flow patterns are identified and resulting implications for performance improvement are elaborated on.

Implement an application and framework (Chapter 5)

Software is implemented which creates a process model of combined historical data. To test performance of a business process using recorded data of other organizations, a software component is created which is able to receive requests and response with advices based on the data in this process model. Furthermore, a framework is designed which enables business processes to communicate with this application. Moreover, software is implemented that stores and calculates performance-related results and statistics.

Construct and perform experiments (Chapter 6)

A business process is chosen from which similar processes are constructed that contain different control-flow pattern configurations. These processes incorporate the framework and serve as input for experiments using a varying set of parameters from which performance is measured and compared.

Conclude based on findings (Chapter 7)

Finally, the results of the experiments are interpreted to conclude the effects of control-flow configuration patterns regarding performance and solve the research questions. Moreover, possibilities for future research are indicated.

1.4 Outline

This remainder of this thesis is structured as follows. In Chapter 2, related work is presented. The broader context of process-aware information systems, process models and event logs are elaborated on. Subsequently, in Chapter 3, the Operational Support Framework is introduced. Recommendations are explained in detail before the way a process model currently is used is described. Next, the changes which are made to make it suitable for this research are given. Chapter 4 addresses configurable process models and the different configurable control-flow patterns these models can contain. Possible resulting implications for recommendations are elaborated on. Then, Chapter 5 discusses the implementation by describing the software used and created for obtaining historic data, process models and recommendation requests and responses. Next, in Chapter 6 an experimental setup is described which decides about various parameters of relevant dimensions in order to obtain a suitable and realistic set of experiments for this research. Then, for these experiments a process model is introduced from which similar variants are constructed that contain different control-flow configuration patterns. These variations among the models are presented in the subsequent section. The results of the experiments are listed in the final section of this chapter. Finally, Chapter 7 presents a discussion on the results and draws conclusions towards answering the research questions. Furthermore, possibilities for future work are indicated.

Chapter 2
Related Work

The preceding chapter explained the problem this research tries to solve by answering the research questions. This chapter describes work that is related to this research. First, process-aware information systems are elaborated on. Then, in the next section, process models are described. Finally, Section 2.3 explains the notion of event logs.

2.1 Process-Aware Information Systems

An information system is defined as a particular type of a work system that processes information. It does so by performing various combinations of six types of operations, which include capturing, transmitting, storing, retrieving, manipulating and displaying information [2]. A work system itself is referred to as a system in which human participants and/or machines perform a business process using information, technology and other resources to produce products and/or services for internal and external costumers [2].

A subset of information systems is called Process-Aware Information Systems (PAIS). PAIS are defined as software systems that manage and execute operational processes involving people, applications and/or information sources on the basis of process models [2]. Process models typically represent business processes by making use of some kind of (graphical) notation. Process models describe business processes in terms of activities and possibly sub processes. By describing causal dependencies, the ordering of these activities is modelled. Furthermore, process models can include descriptions of time-related properties and resources that can carry out activities. Also, data on which decisions are based in the process can be specified [3]. These models are further elaborated on in the next section.

The main difference between more traditional information systems and PAIS is the focus of the system. Where information systems take a data-driven approach, focussing purely on the tasks they have to perform, PAIS on the other hand look at the process they have to support. As a consequence, PAIS are able to support organizations with their business process executions by providing insight in the status of both the process as a whole and tasks which are part of it. This enables organizations to monitor and communicate about their current state and performance.

As will become clear in the next sections of this chapter, PAIS is the class of information systems for which this work is relevant.

2.2 Process Models

The previous section described PAIS. PAIS are able to execute business processes based on process models. A process model represents a business processes by making use of a (graphical) notation that can be seen as a blueprint for a set of process instances with a similar structure [2]. In such a model, process entities and the relationships among them are defined. Process models can play an important role in organizations, since they can be used for various purposes, such as [3]:

- Insight and discussion: A process model can cause modellers to view a business process from various angles and trigger discussions about it among stakeholders
- Verification: Process models can be analyzed to find potential problems
- Specification: A process model can be used to specify a system before its actual implementation and therefore serve the role of a 'contract' between a customer and a provider
- Performance analysis: Some process models can be used for analysis of a business process and factors involved, like waiting times, service levels, throughput etc.
- Configuration: Process models can be used to configure a system or business process using PAIS

There is a wide range of business process languages and notations, such as transition systems, UML, EPC, BPMN and Petri nets. In this thesis, transition systems and Petri nets are used, because they are the most suitable for this work. There are several reasons for this choice. First of all, transitions systems are chosen, because any process model with executable semantics can be transformed into such a model [4]. Moreover, Petri nets not only provide an intuitive way of representing a business process, but also enable the use of many analysis techniques, caused by its formal semantics.

A transition system is a process modelling notation consisting of states and transitions [4]. Each state has a unique identifier and can be connected with other states by a transition. A transition has the label of an activity and represents the movement of a system from a state to a certain other state by carrying out that corresponding activity.

Figure 2.1 shows an example of a (small) transition system depicting a business process. In this process, either activity 'A' or 'E' is carried out first. If activity 'E' is executed, 'F' will be carried out next and the process is finished. If activity 'A' is carried out, a choice has to be made whether activity 'B' or 'C' should be executed next. Then, for both choices, the execution of activity 'D' finishes the process.

Note that the state drawn with a dotted line is the initial state of the process. An initial state only has outgoing transitions. The states with thick lines are final states, indicating the end of the process. A final state only has incoming transitions.

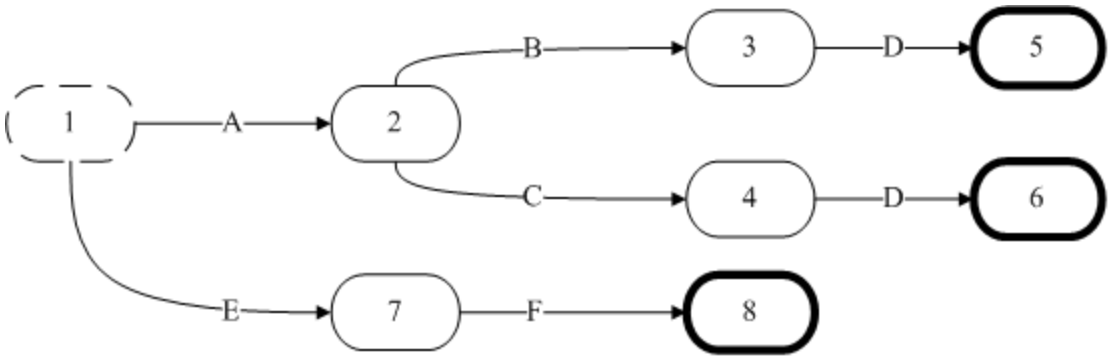

Figure 2.1: Example transition system

The business process described can also be modelled as a Petri net. Figure 2.2 shows a Petri net for the same business process as of the given transition system. Unlike a transition system, a Petri net is not a process modelling notation, but a process modelling language. It is a bipartite graph that consists of places and transitions [4]. The state of a Petri net is called its marking and determined by the distribution of tokens over the places of the net. The token in Figure 2.2 indicates that the process is at the initial state, which is equal to being in state 1 in the transition system of Figure 2.1. A Petri net moves from one state to another by firing a transition, which consumes tokens on places connected to the transition with an input arc, and produces tokens on places connected with an output arc. For instance, firing transition 'A' will remove the token and produce one in the place having an incoming arc from that transition.

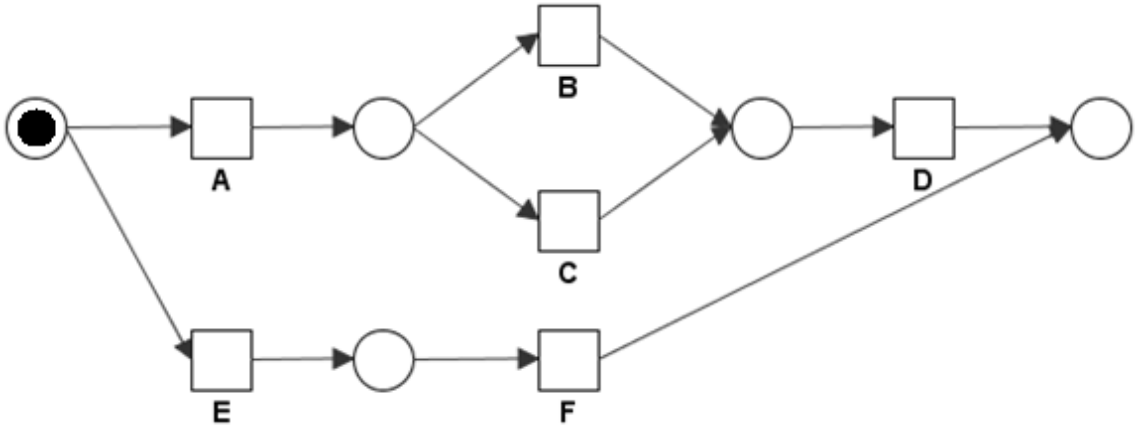

Figure 2.2: Example Petri net

Petri nets, also referred to as classical Petri nets, can be extended to Coloured Petri nets to enlarge its functionalities. Classical Petri nets have the limitations that they cannot express time and are not able to differentiate between tokens in the same place. Moreover, classical petri nets cannot model the hierarchical structure of a process. Therefore, these nets are extended with colour, time and hierarchy.

Figure 2.3 shows the Petri net of Figure 2.2 extended with colour and time. Again, the same business process is depicted, but this time more information is included in the process model. The values of the attributes of tokens are used to distinguish tokens from each other. In the example, the value of attribute 'ID' is a unique identifier. Transitions can change the values of attributes. As can be seen in Figure 2.3, the token with 'ID=3' has a score of 10 after firing transition 'A'.

To indicate how long an activity of a business process takes, time is added to a Coloured Petri net. A token in such a model has a timestamp, which represents the moment it is available for possible firings of transitions. A transition is enabled, in other words can fire, at the time at which all its input places contain available tokens and there is no other transition with a smaller enabling time. Transitions produce tokens on output places with a timestamp that equals the time of firing increased with a possible delay. This delay models the duration of activities in a business process.

The Coloured Petri net in Figure 2.3 shows that firing transition 'A' added a delay of 5 time units to the token with attribute 'ID=3'. Additionally, firing 'E' produced a token with timestamp '2'. At this moment, only transitions 'A' and 'E' are enabled, since they have the smallest enabling time.

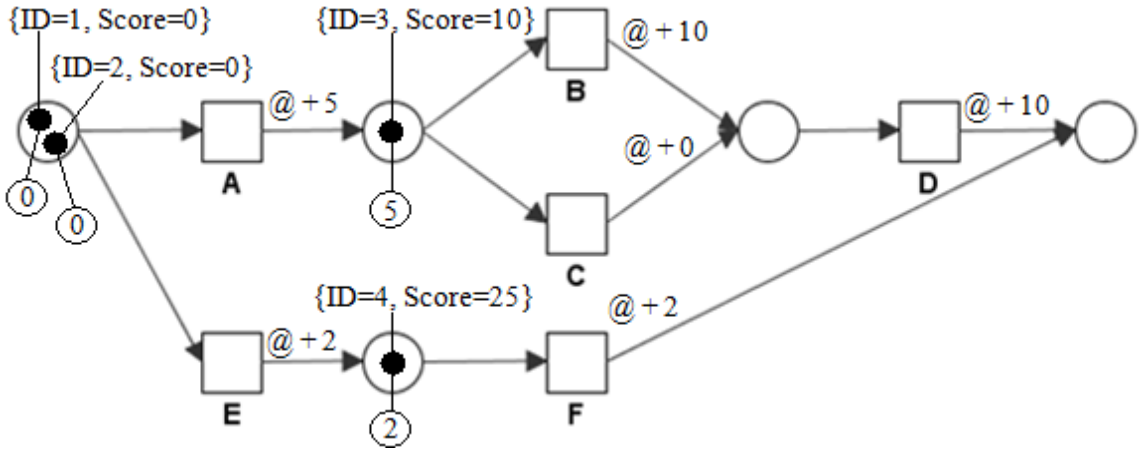

Figure 2.3: Example Coloured Petri net

Note that hierarchy is not included in the example, because of its small size. Although with colour and time a business process can be modelled, this can still result in an unmanageable process model if that business process is very large. Hierarchy makes it possible to reflect the hierarchical structure of a business process in a Petri net. In Chapter 6, a Coloured Petri net of a business process is shown that contains this third extension of classical Petri nets.

Typically, there are different implementations of Coloured Petri nets. One of them is CPN ML [5], which enriches the possibilities of (classical) Petri nets with colour, time and hierarchy by making use of the high-level programming language CPN ML. This implementation is extensively used for this research and is further elaborated on in Chapter 5.

2.3 Event Logs

Process-aware information systems support the execution of business processes. The processes that are carried out by these information systems typically are stored in so called event logs. An event log is a recording of one or multiple processes [6]. Table 2.1 depicts a fragment of an example event log.

As Table 2.1 shows, each record in an event log contains a case identifier, the name of the corresponding event, an event type indicating the current state of that event, the resource that carried it out and a timestamp showing the logging time. Moreover, optionally data fields can log additional information.

In an event log, for every process, zero or more cases - also called process instances - are recorded. The partial event log in Table 2.1 shows two different cases. Furthermore, for each case, zero or more events are included. An event is the recorded execution of a task or activity. Each event should have at least a name and an event type. Table 2.1 lists six events for case 1 and two events for case 2. For instance, at 11:10 the resource 'trainee' started writing a short letter for case '1'. At 12:30, he/she completed this task.

Case	Event	Event Type	Resource	Time
1	Receive request	Start	Administrator	09:00
1	Receive request	Complete	Administrator	10:30
1	Check content	Start	Administrator	10:30
2	Receive request	Start	Trainee	10:30
2	Receive request	Complete	Trainee	11:00
1	Write short letter	Start	Trainee	11:10
1	Write short letter	Complete	Trainee	12:30
1	Check content	Complete	Administrator	13:00
...				

Table 2.1: Example event log

Event logs are used as input for various process mining techniques. Because there are a number of software tools which implement these techniques, initiatives were started to define a common format for event logs. Through this initiative the MXML, and later the XES format, were constructed [7]. Event logs in both these formats can be used by the process mining software tool ProM [6], which is extensively used throughout this project and described in Section 5.1.

Chapter 3

Operational Support

In the previous chapter, related work to this master project was presented. In this chapter we first discuss an approach which is able to support running executions of a business process. A technique that tries to improve performance by making use of own historical data then is explained in the second section. Next, the use of a transition system for this purpose is described. Finally, the additions to this process model that are made to make it suitable for this research are explained in Section 3.4.

3.1 Operational Support Framework

As mentioned in Chapter 1, process mining is a way to extract knowledge from systems by taking event logs as a starting point. Process mining is applied in both an offline and online setting [8]. In an offline setting, both partial and completed cases can be taken into consideration for offline analysis. For instance, analyzing data could eventually lead to modifying or extending a process model. Although this can indirectly lead to different behaviour of partial, still running cases, this cannot be interpreted as directly intervening in their executions. In an online setting, on the other hand, process mining is used in an active manner to directly influence a running case. This type of process mining is also called Operational Support and consists of three types of actions, namely checking, predicting and recommending, which are illustrated in Figure 3.1.

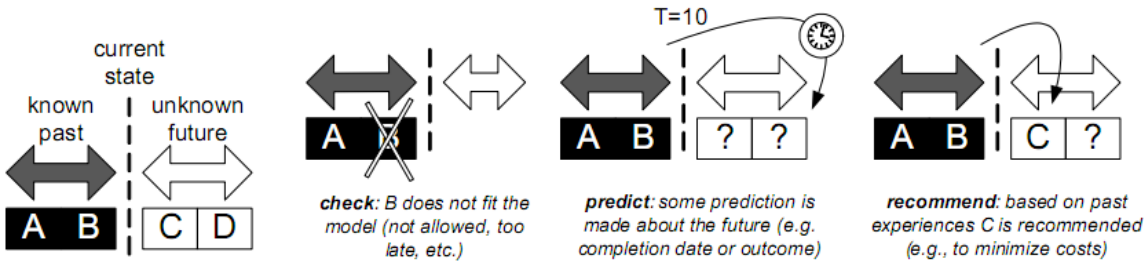

Figure 3.1: The three different types of Operational Support [8]

Checking projects a running case, also referred to as a partial trace, on a process model to see if the current execution is valid. Based on the fit, notifications or alerts can be generated. In Figure 3.1, executed activity 'B' of the depicted current state of a partial trace [A, B] cannot be mapped onto model.

Prediction provides an estimation of the further development of a partial trace based on the information of comparable, already completed, cases in the event logs. A prediction can for instance involve information about the expected remaining time of the case, costs and probabilities of an outcome.

Recommendations furthermore suggest the next action which should be taken that leads to the best (predicted) performance in terms of time, quality or costs of the running case.

Using the online and active technique of recommendations, it will be investigated if performance can be improved by making use of historical data of other organizations. Therefore, a detailed view of the current state of this notion in the process mining field is provided in the next section.

3.2 Recommendations

Recommendations apply process mining on-the-fly by looking at an event log (a set of completed executions) and an active, current partial execution. Predictions about the future of that partial case are made using the history inside the event log. This leads to giving recommendations: advices on which activity to carry out next for a case, given a certain objective, such as reducing process times. Figure 3.2 shows this process visually.

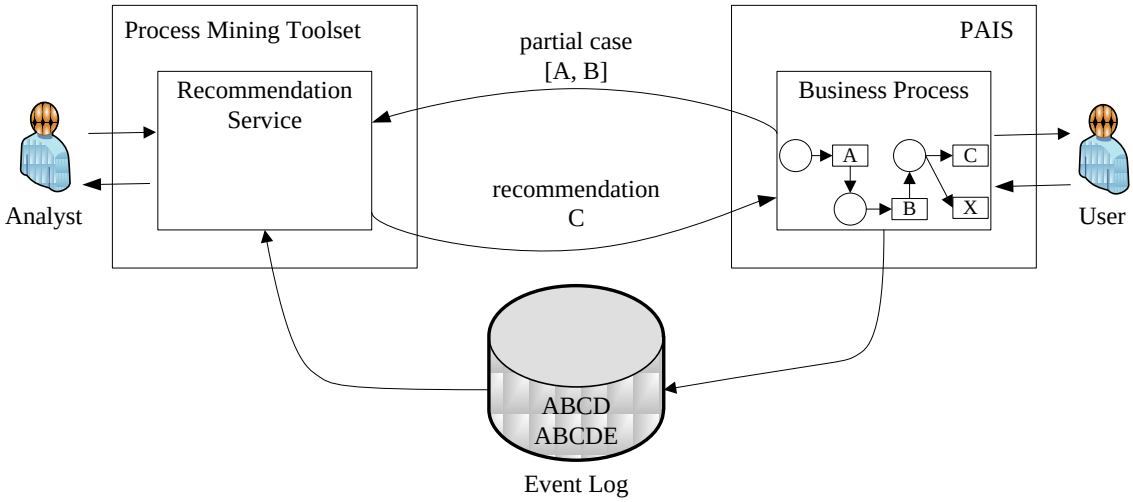

Figure 3.2: Recommendation framework adapted from [9]

A recommendation service uses an event log to base recommendations on by comparing the partial case with the already completed cases present in the log. Because in principle no two cases are dealing with exactly the same data, some sort of abstraction is necessary. There are several abstraction mechanisms with which cases can be compared to obtain a degree of similarity [9]:

- Finite or infinite horizons: take the entire case or only the last *n* events. The horizon can also be expressed in time (consider the activities of last week) or other data objects.
- Filtering of events: only a selected set of events is considered. Certain events or event types are removed.
- Remove order and/or frequency. There are three ways of representing cases:
  - Sequence: both the order and frequency of events is considered for a particular case
  - Multi-set (or bag) of events: the order of events will be lost, but the frequency each event is executed is preserved
  - Set of events: both the order and frequency of events will be lost; only the presence of events is taken into consideration

Examples of the possible abstractions presented above are depicted in Table 3.1 for a partial case 'ABCBDDCE'.

Case	**Infinite**	**Finite (n = 4)**
	ABCBDDCE	**DDCE**
Sequence	<A, B, C, B, D, D, C, E>	<D, D, C, E>
Multi-set	{A, B^2, C^2, D^2, E}	{C, D^2, E}
Set	{A, B, C, D, E}	{C, D, E}
Filtered sequence (B, C)	<A, D, D, E>	<A, D, D, E>
Filtered multi-set (B, C)	{A, D^2, E}	{A, D^2, E}
Filtered set (B, C)	{A, D, E}	{A, D, E}

Table 3.1: Possible abstractions of example case

As can be seen in Table 3.1, various combinations of abstraction techniques can be applied on a partial trace. The second column shows abstractions of the partial trace using an infinite horizon. The third column only takes the last 4 events into account. Additionally, events 'B' and 'C' are filtered out in the examples at the last three rows.

The abstraction mechanisms can also be used on completed cases in an event log. This historical data can be used to build a transition system, possibly annotated with time [8]. A running case can then be mapped to this transition system to determine the current state of this partial trace and the activity that should be recommended, possibly based on time information. This is shown in the next section.

3.3 Use of a Transition System

Recall that any process model with executable semantics can be transformed into a transition system. As discussed in the previous section, to provide recommendations, current partial instances of a business process have to be compared with historical data.

First, a transition system is derived from historic data. Then, this process model is navigated by taking the paths with transitions that equal the carried out activities of the process instance. In such a way, the current state of the partial execution is determined. Then, depending on the outgoing transitions of this current state in the graph, the activity which corresponds to a transition can be recommended.

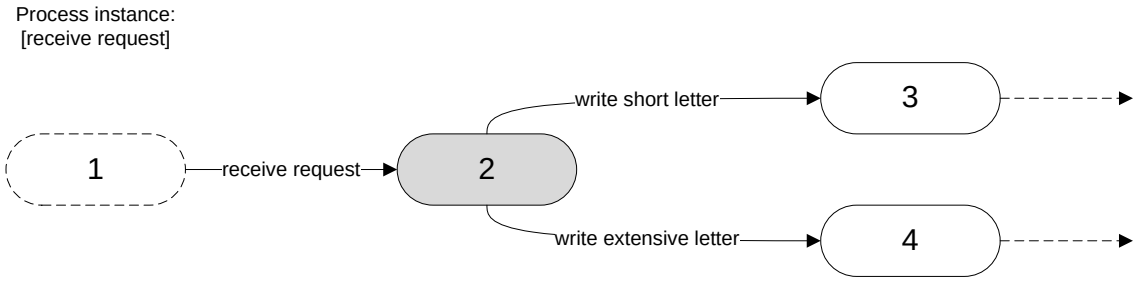

Figure 3.3: Transition system

Figure 3.3 shows an example of this. If state '1' would be the initial state in the transition system and the current process instance contains only activity 'receive request', the current state of the process instance would be state '2'. From this state, there are two transitions that change the current state into another state. However, it is obvious that only knowledge of the labels of the activities is not enough for providing a reasonable recommendation about which activity to do next ('write short letter' or 'write extensive letter'). Clearly, making use of a transition system only is effective when more information is available.

As described in Section 2.3, records in event logs contain a timestamp. The transition system can thus be annotated with time information [8]. This allows for computation of various timing information, including minimal, maximal and average flow, elapsed and remaining times of cases that once were in that particular state. Returning to the example of Figure 3.3, when states '3' and '4' contain this information, a decision about which activity to recommend can be made with more certainty, see Figure 3.4.

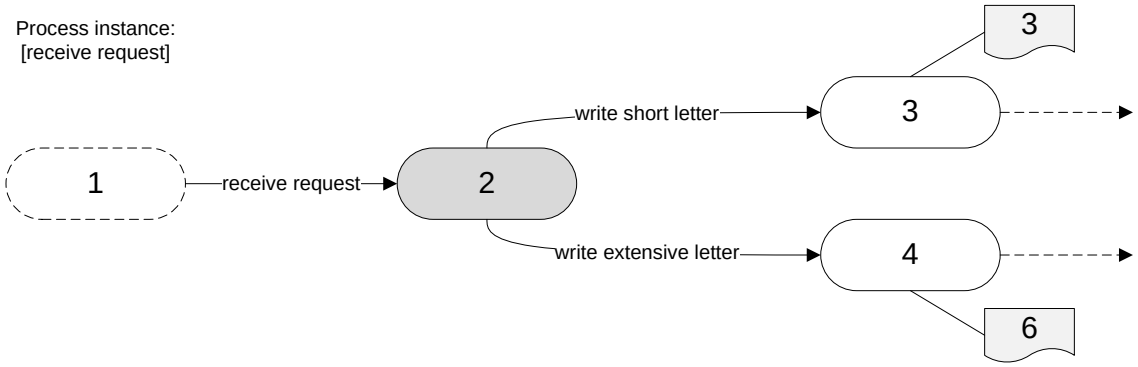

Figure 3.4: Annotated transition system

If the transition system would contain remaining time 3 units for state ‘3’ and remaining time 6 units for state ‘4’, the activity ‘write short letter’ is recommended if the objective is achieving shortest process time.

Although the transition system in Figure 3.4 now contains information so that a reasonable recommendation can be given, it is not sufficient for this research. The reason for this is the fact that data of multiple organizations has to be included in the transition system. The approaches to solve these shortcomings and possible scenarios that can arise are explained in the next section.

3.4 Transition System on Multiple Event Logs

The previous section first reasoned that only the presence of states and transitions connecting these states is insufficient in order to be useful for recommendations. Afterwards, it was shown that by making use of time information, more accurate recommendations can be provided. This section describes the additional steps required to improve the usefulness of transition systems for this research.

In the current state of the research field, the relevant historical data is stored in one event log. From that event log, a transition system is then derived.

However, for this work, historical data of multiple organizations has to be used to base recommendations on. This data of organizations 1...*n* is recorded in event logs 1...*n*. This new arised situation results in constructing transition systems 1...*n* from event logs 1...*n*. Figure 3.5 sketches this new situation.

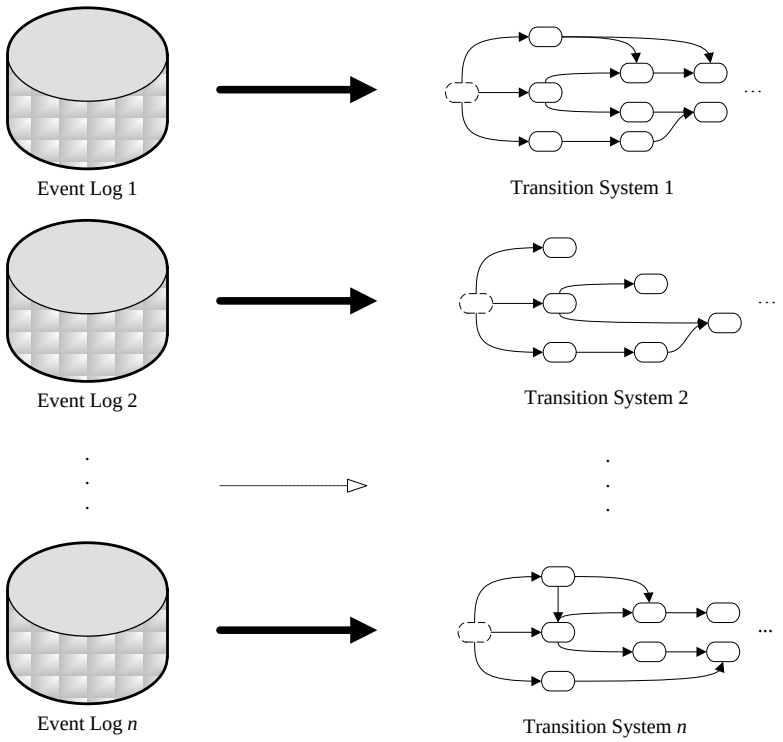

Figure 3.5: Constructing transition systems 1...*n* from event logs of organizations 1...*n*

However, this situation is not desired for this research. Although the transition systems can be used to try to fit a partial trace of a business process of organization *i* with a path on a transition system *t*, it makes more sense to somehow again obtain one transition system. Note that the approach of Figure 3.5 could work by creating mappings among possible similar states of transition systems with respect to their incoming and outgoing transitions. However, an easier solution is achieved by combining the multiple transition systems. In this approach, actually not the transition systems themselves are merged, but the event logs instead. That is, event logs 1...*n* are merged together in order to obtain a so called collective event log, containing all historical data of organizations 1...*n*. All traces of all event logs are included in this collective event log. It is not needed to preprocess the logs by for instance removing duplicates or worse performing traces, because typically you want to have the availability of as much historical data as possible to derive recommendations from. This is especially the case, since the recommendations will be based on average process times of specific behaviour.

From the obtained collective event log, one transition system is constructed. This transition system is in the remainder of this thesis referred to as the collective transition system and contains all historical data of organizations 1...*n*. The above described process is visualized in Figure 3.6.

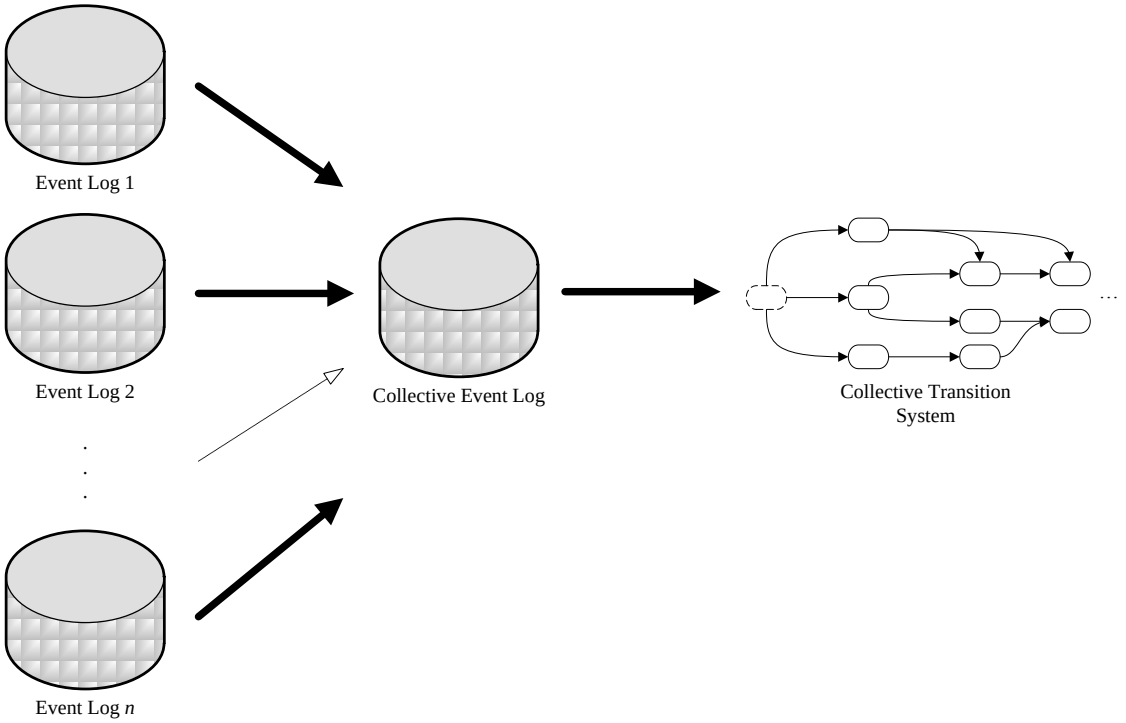

Figure 3.6: Constructing a collective transition system from a collective event log

The collective transition system can now be used to compare partial traces of one organization with historical data of other organizations. Note that compared to the existing situation in literature, in which a transition system is derived from a single event log which originates from the same organization, other scenarios can occur.

First of all, it is likely that the situation in which no state in the transition system can be found which corresponds to the partial trace for which a recommendation is requested will occur more often. For instance, take [receive request, check content] as the partial trace of an organization. Further, let the collective transition system contain the states, transitions and time annotations shown in Figure 3.4. In this scenario, the partial trace cannot be mapped to any state.

A second problem relates to the actual recommended activity. Again, the same collective transition system as in the previous example is used, but now the partial trace is [receive request]. The activity 'write short letter' will be recommended (because it connects with the state with the least remaining time). However, it could be the case that this activity currently cannot be executed. In fact, it even is possible that this activity is not included in the business process of the present organization.

Constructing a collective transition system will only make sense when the corresponding organizations are similar. A number of conditions have to be satisfied, because otherwise the collective transition system containing historical data of organizations 1...*n* will consist of *n* separate parts.

This chapter discussed the recommendations technique of the operational support framework. Recommendations make use of historical data in order to improve performance. This addresses sub question 1. Moreover, an approach was introduced that combines data of multiple organizations such that it can be used for performance improvements. With this, sub question 2 is also covered.

The next chapter defines similarity of organizations and corresponding requirements. Different configurations of control-flow patterns are elaborated on. Furthermore, implications of these configurations for recommendations are described. Then, Chapter 5 deals with the scenarios described above. Solutions to these problems are presented, as well as the implementation of this research.

Chapter 4

Similar Organizations

After the description of related work in Chapter 2, the previous chapter elaborated on an approach which is able to improve performance of running cases by making use of historical data. Furthermore, the process model that is used for this purpose was given. At the same time, limitations of this model for this research were indicated. An extended version of this process model was explained which solves these and can be used for historical data of other, similar organizations.

This chapter discusses the notion of similarity of organizations. Configurable process models and their possible control-flow patterns are introduced. Moreover, possible impacts of control-flow pattern configurations on the usability of recommendations are presented.

4.1 Configurable Process Models

Configurable process models are a specific type of process models. One of the capabilities of these models is the potential acceleration of the modelling of a business process. This is achieved by the fact that typically, configurable process models provide a collection of generic solutions of relevant models [10]. In other words, configurable process models integrate the different variants of a business process into a single model [11]. By configuring the model, a process model optimally fitting an organization's individual needs is derived.

For this master project, a configurable process model is used to create different variants of a process model. These variants represent organizations that carry out a business process in a different way.

Making use of configurable process models comes with a number of advantages. As discussed, a first obvious advantage is the fact that a model is easily adapted to a particular organization. That is, the configurable process model expressing the superset of models can be reduced to the subset of models relevant for a particular organization. A second benefit of using configurable process models is that by deriving a variant, you stay in the previously defined solution space of the superset of models. This implies that possible errors in the resulting process models are avoided. Additionally, configurable process models decompose a model into manageable pieces, which enables easily adaption needed through changes in demands [12]. Finally, usage of configurable process models also results in identical semantics for tasks or

activities. Especially this aspect is beneficial for this research, because the organizations will make use of the same vocabulary and therefore information retrieval related techniques with respect to naming conventions do not have to be used.

In a configurable process model, the following configuration patterns for the control-flow dimension are available:

- Optionality: choose to either execute or skip an activity [12]
- Blocking: choose to either block (not execute) or not block (execute) an activity [13]
- Hiding: choose to hide or not to hide an activity [13]
- Interrelationship: execution of an activity depends on the execution of another activity [12]
- Interleaved parallel routing: choose in which order a set of activities is executed [12]
- Parallel split: choose a subset of the activities [12]
- Exclusive choice: choose exclusively an activity [12]
- Multi-choice: any possible combination of activities [12]
- Aggregation: combine activities into one activity [14]

These patterns are explained in detail in the next section.

4.2 Control-Flow Configuration Patterns

This section describes the various control-flow patterns that can be configured in a configurable process model. For each pattern, multiple switches are available. These can be turned on, off or optional and imply the routing of the resulting process model. By turning switches on or off, the configuration of the corresponding control-flow pattern is decided at configuration-time. This means that all process instances of the model will contain exact the same behaviour as the setting specifies. When a control-flow pattern is configured as optional, the decision about the execution of the corresponding activities is made in the process model, which implies that in each process instance a separate decision can be made. This is called a run-time decision.

4.2.1 Optionality

There are three different configurations for the optionality control-flow pattern, on, off (configuration-time) and optional (run-time):

- **On** All process instances of a process model contain the corresponding activity
- **Off** No process instances of a process model contain the corresponding activity
- **Optional** The occurrence of the corresponding activity differs per process instance of the process model

4.2.2 Blocking

Blocking specifies that an activity cannot be executed. Blocking is a special control-flow pattern, in terms of its scope. In contrast to other patterns, blocking can also have a direct impact on subsequent activities not part of the pattern. For instance, blocking activity A*m* has as a consequence that activities A*m*+1 to A*n* also cannot be executed anymore when these activities would be located in a sequential order in the same branch.

The following (configuration-time) switches are applicable for this work:

- **On** None of the process instances of a process model contain the corresponding activity, and possibly the activities which are blocked as a result of the applied instance
- **Off** All process instances of a process model contain the corresponding activity

4.2.3 Hiding

For this research, a hidden activity is an activity that is executed, but not recorded in an event log. There are two (configuration-time) options for this control-flow pattern:

- **On** All process instances of a process model hide the activity
- **Off** None of the process instances of a process model hide the activity

4.2.4 Interrelationship

Interrelationship control-flow patterns are divided into two types: mutually exclusive and mutually dependent.

Mutually exclusive specifies that activities have to be configured opposite to each other. There are three switches for activities ‘A’ and ‘B’:

- **On, off** All process instances of a process model only execute activity ‘A’
- **Off, on** All process instances of a process model only execute activity ‘B’
- **Optional** The choice of executing activity ‘A’ or ‘B’ is made separately for each process instance of the process model

The mutually dependent interrelationship pattern, on the other hand, specifies that corresponding activities must be handled in a consistent way. This results in the following configuration options:

- **On** All process instances of a process model execute both activities ‘A’ and ‘B’
- **Off** All process instances of a process model do not execute activities ‘A’ and ‘B’
- **Optional** The choice of either executing both activities ‘A’ and ‘B’ or none of them is made separately for each process instance of the process model

4.2.5 Interleaved Parallel Routing

The execution orders for a number of activities are configured using the interleaved parallel routing control-flow pattern. Activities for which their mutual ordering has semantically no consequences on the process can be ordered in *n*! possible ways, see Figure 4.1. This ordering can be decided:

- **Globally** All process instances of a process model have the same ordering of *n* activities (configuration-time)
- **Locally** Per process instance, the ordering of *n* activities is decided separately at run-time

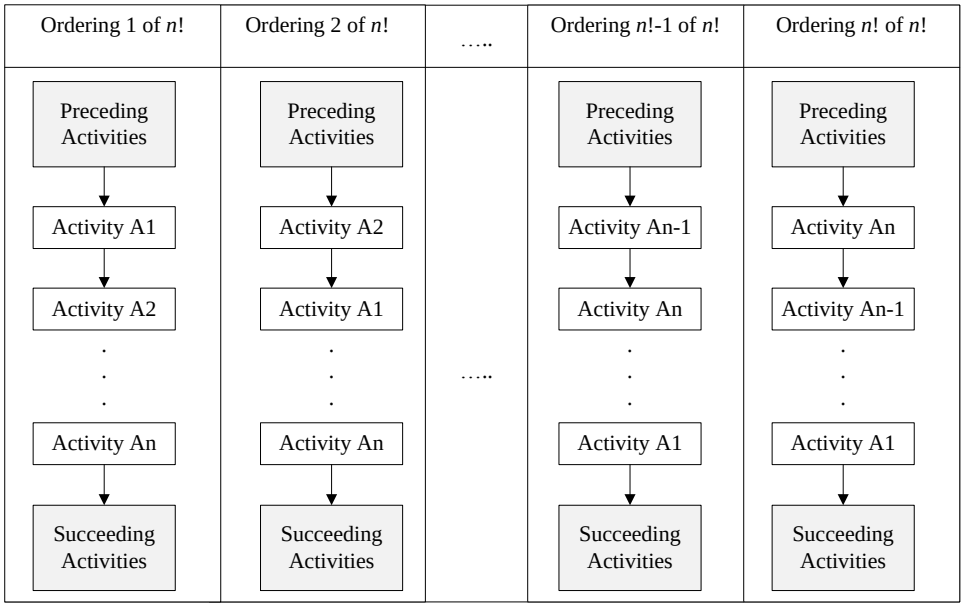

Figure 4.1: Ordering possibilities of *n* activities of interleaved parallel routing pattern adapted from [12]

4.2.6 Parallel Split

The parallel split control-flow pattern specifies which *m* of *n* activities, where *m* <= *n*, have to be carried out. These *m* activities can be executed concurrently. Thus, a parallel split pattern containing *n* activities has configuration options which turn switches of activities A1 to A*m* on. When *m* < *n*, activities A*m*+1 to A*n* are switched off. These decisions are made at configuration-time.

4.2.7 Exclusive Choice

There are *n* different configuration options for an exclusive choice control-flow pattern which contains *n* activities. For each option, a different activity is switched on. All other activities are

switched off. These decisions are taken locally per process instance, thus at run-time. Note that configuring such a pattern at configuration-time would be the same as turning all other activities off, see the optionality control-flow pattern in Paragraph 4.2.1.

4.2.8 Multi Choice

The multi choice control-flow pattern can specify that any combination of *n* activities have to be executed. This behaviour can already be captured using optionality, parallel split and exclusive choice patterns. Therefore, its configuration options are omitted.

4.2.9 Aggregation

The aggregation control-flow pattern indicates a hierarchical level of an activity. Multiple activities combined semantically represent one activity. This is visualized in Figure 4.2. For instance, activities ‘A11’ and ‘A12’ together represent activity ‘A1’. An activity A*m* can be divided into sub activities A*m*1 to A*mn*. As such, there are three types of configuration options:

- **On** In all process instances of a process model, the activity A*m* is split into activities A*m*1 to A*mn* making use of aggregation level *n* (configuration-time)
- **Off** Aggregation is not applied to the activity in all process instances of a process model, which corresponds to the setting ‘on’ of the control-flow pattern ‘optionality’ (configuration-time)
- **Optional** Each process instance of a process model can choose a different aggregation level *n* (run-time)

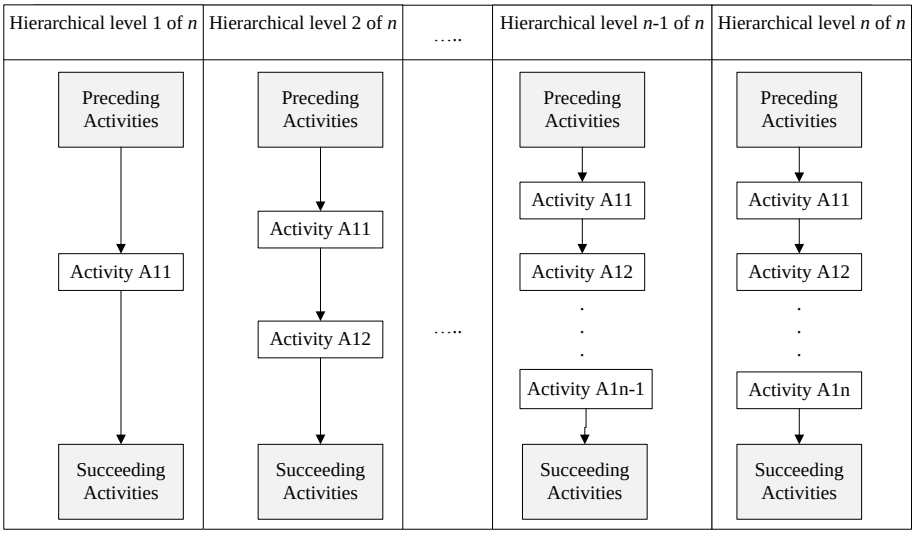

Figure 4.2: Aggregation possibilities *n* for activities

4.3 Implications for recommendations

The previous section discussed the different control-flow patterns which can be part of a configurable process model and their corresponding configuration switches. This section elaborates on implications of these patterns for this work.

The behaviour of process models is recorded in event logs. These event logs are used to create a collective transition system, which can be used for providing recommendations to (other) organizations. Because these organizations possibly contain varying control-flow configuration patterns than the organizations from which the event logs were used, various scenarios can arise. These scenarios and the fact whether they provide problems are discussed below.

- **Optionality**
  - *Activity is not contained in the collective event log, but can occur in an organization*
    As a consequence, the activity cannot be recommended because it is not part of the collective transition system. The implementation, which is discussed in the next chapter, elaborates on how this problem is solved. It is visualized in Figure 4.3.

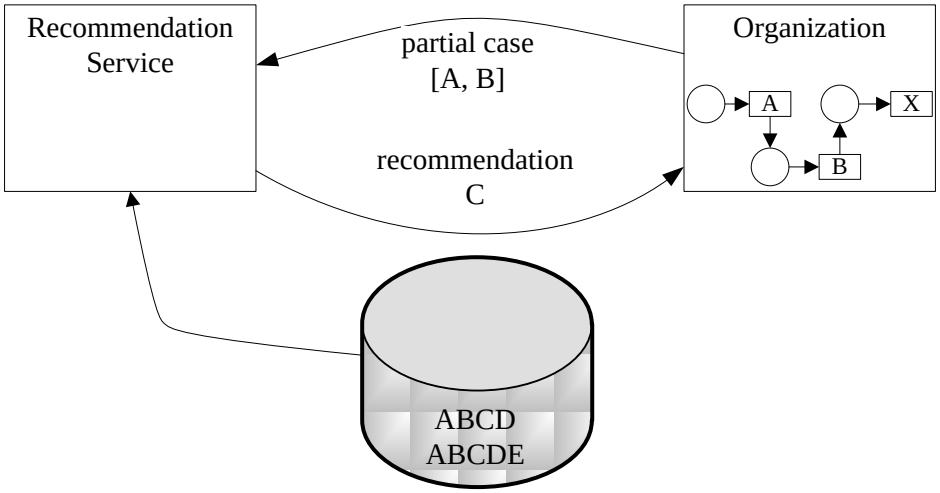

Figure 4.3: Activity can occur in organization, but is not contained in the collective event log

  - *Activity is contained in an event log, but cannot happen in an organization*
    This potentially is a problem, because recommending an activity which cannot be carried out in an executable process model causes a deadlock. Furthermore, the partial trace of the organization will in a lot of cases not fit a state in the collective transition system, because the activity is part of the paths from the initial state in the collective transition system to another state, but not of the trace. Figure 4.4 depicts this scenario.

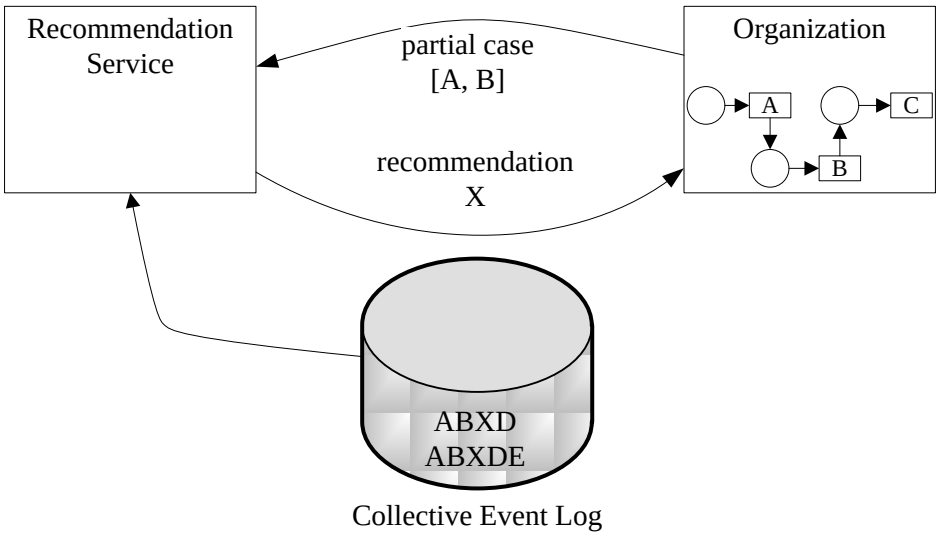

Figure 4.4: Activity is contained in the collective event log, but cannot occur in organization

- **Blocking**
  When in an organization a blocking pattern is configured as on, the corresponding activity (and possibly subsequent activities) cannot be executed. This can result in the scenario shown in Figure 4.4. The scenario in Figure 4.3 can only occur if the collective transition system is constructed from event logs of organizations that all configured blocking for the same activity as on, and the organization which requests a recommendation configures the blocking pattern for this activity as off.

- **Hiding**
  When a hidden activity is configured as on, process instances do not record it. Therefore, the collective transition system does not contain this activity. If in the organization which requests a recommendation this activity occurs, the problem depicted in Figure 4.3 can arise.

- **Different interrelationship patterns of activities**
  - For the mutually exclusive pattern, there only is a problem when all process instances in the collective event log, and thus collective transition system, carried out the same activity and the organization which requests a recommendation executes the other activity of the pattern. It is most likely that this situation will not occur. If it does, the scenarios of optionality of an activity can occur.
  - For the mutually dependent pattern, a problem arises when the collective transition system does not contain the activities that do occur together in the organization which requests recommendations. Vice versa, when both activities occur in all traces in the event log and not in the model of the organization, again, problems with the fit can occur.

- **Interleaved parallel routing pattern of activities**
  The different ordering of activities is only problematic if the sequence abstraction technique is applicable. Recall that for this technique, no abstraction of the ordering of activities is applied. As a consequence, different ordering results in the fact that the partial trace of the organization cannot fit a path in the collective transition system, and therefore a reasonable recommendation cannot be provided.

- **Parallel split of activities**
  This control-flow pattern can cause several problems, because it can impact both ordering and choice of activities. Therefore, it is a mix of the earlier indicated problems of other configurations of the optionality and interleaved parallel routing patterns.

- **Exclusive choice**
  See the scenarios of the optionality pattern

- **Aggregation**
  A different level of aggregation in an organization compared with the historical data in the collective event log causes similar problems as mentioned for optionality, because the mismatch in activity sets cannot be recognized.

The previous chapter described that by making use of recommendations, performance of organizations can be improved using historical data. Furthermore, an approach was introduced such that historical data of multiple organizations can be used for performance improvements. This covered sub questions 1 and 2.

This chapter discussed configurable process models. Their characteristics and possible control-flow configuration patterns were explained. Also, the implications on recommendations caused by different configurations of control-flow were stated. Therefore, sub question 3 is addressed.

The next chapter presents the implementation of the approach described in Chapter 3. Implementations of solutions to the problems mentioned there and in this section are given. Moreover, software components are elaborated on which enable the simulation of experiments, such that sub question 4 can be answered.

Chapter 5

Implementation

In the preceding chapters, an approach was discussed which is able to improve an organization's performance based on its own historical data. Additionally, required steps in order to make the approach suitable for this research were introduced. Moreover, Section 3.4 and Chapter 4 elaborated on various issues that can arise by making use of cross-organizational data for recommendations. This chapter explains the implementation of the described approach and solutions to these potential problems. Before presenting the implementation solution, first a high-level overview of the required components for this research is shown. This high-level overview of the architecture places each component in the entire context and is depicted in Figure 5.1.

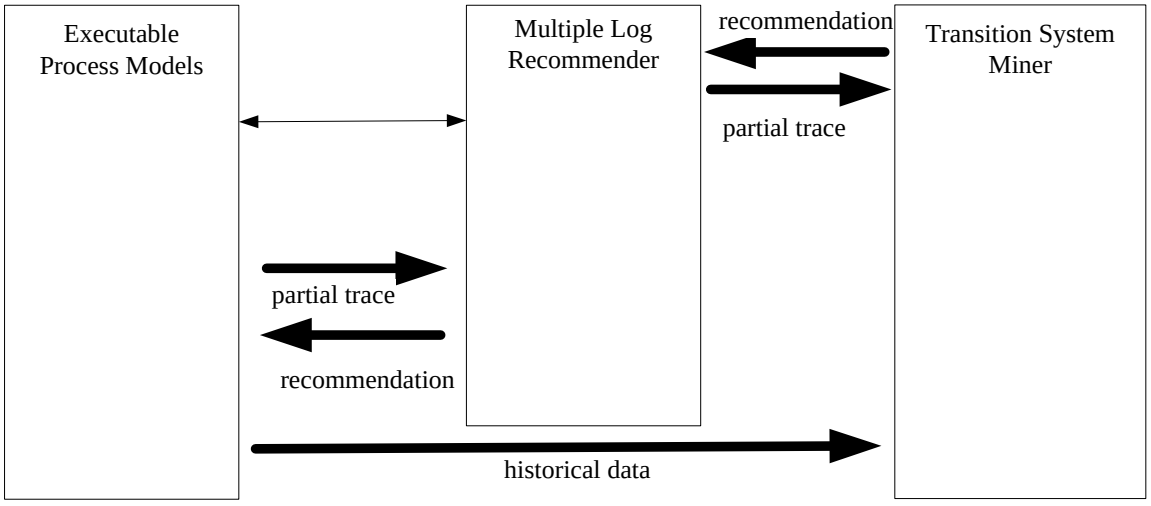

Figure 5.1: High-level architecture of the implementation

Figure 5.1 shows the high-level architecture of this research. Historical data of executable process models is used as input for the Transition System Miner. In other words, from event logs 1...*n* of organizations 1...*n*, a collective transition system is constructed. This implementation is described in Section 5.1.

The event logs are obtained by executing process models of organizations in a PAIS. Furthermore, process instances of models describing these organizations have to communicate with a service to request and receive recommendations. As is shown in Section 5.2, this is done in

such a way that a lot of the problems given in Sections 3.4 and 4.3 are solved. Besides the communication of recommendations, the model processes the recommended activities as well. Section 5.2 elaborates on executable models which perform both these tasks.

The service which determines the activity to be carried out next for a partial case is called the 'Multiple Log Recommender'. As the implementation of this application explains in Section 5.3, this recommendation service takes three collective transition systems and information of a process instance as input, to base its advice on. Additional details about the Operational Support Framework that is implemented are given in that section.

Finally, Section 5.4 discusses a component that is implemented to start simulations of business processes of organizations using different historical data, organizational process models and other settings. Moreover, it retrieves and calculates statistics and results of process executions.

5.1 Collective Transition System Miner

The architecture of this implementation corresponds to Figure 3.6 of Section 3.4, which depicts the approach for combining historical data of multiple organizations such that it can be used for this research. This sub architecture is depicted in Figure 5.2.

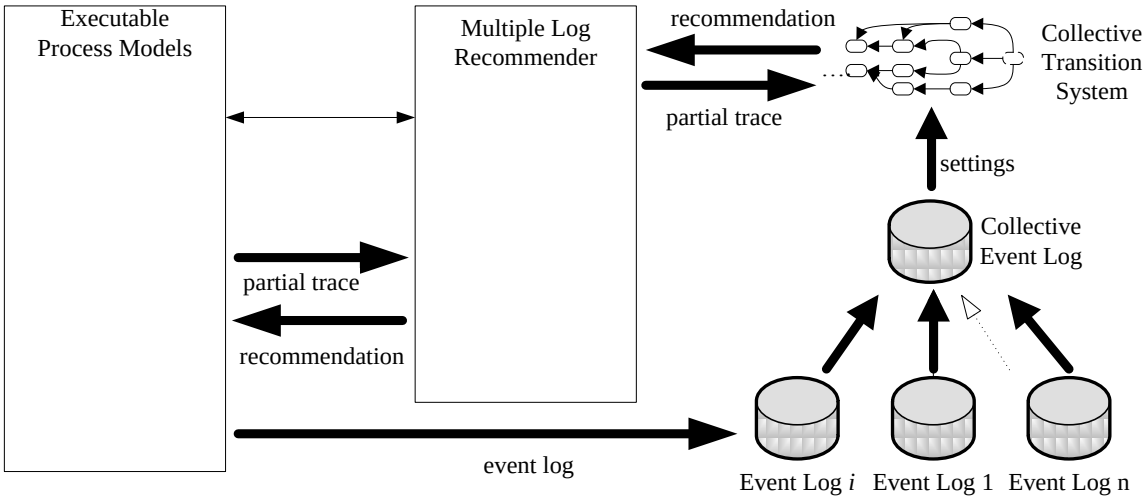

Figure 5.2: Architecture of the implementation highlighting Collective Transition System Miner

The implementation of the collective transition system is realized using the ProM Framework. The ProM framework is a software tool which offers a large number of process mining techniques that provide ways to extract knowledge about a process [15]. ProM[1] is developed by the AIS group, under which this master project is carried out. Because of its implementation in Java, enabling platform independency, and its modularized setup, the tool offers a lot of accessibility and flexibility such that researchers and developers can contribute to the software.

[1] For more information, see http://www.promtools.org/prom6/

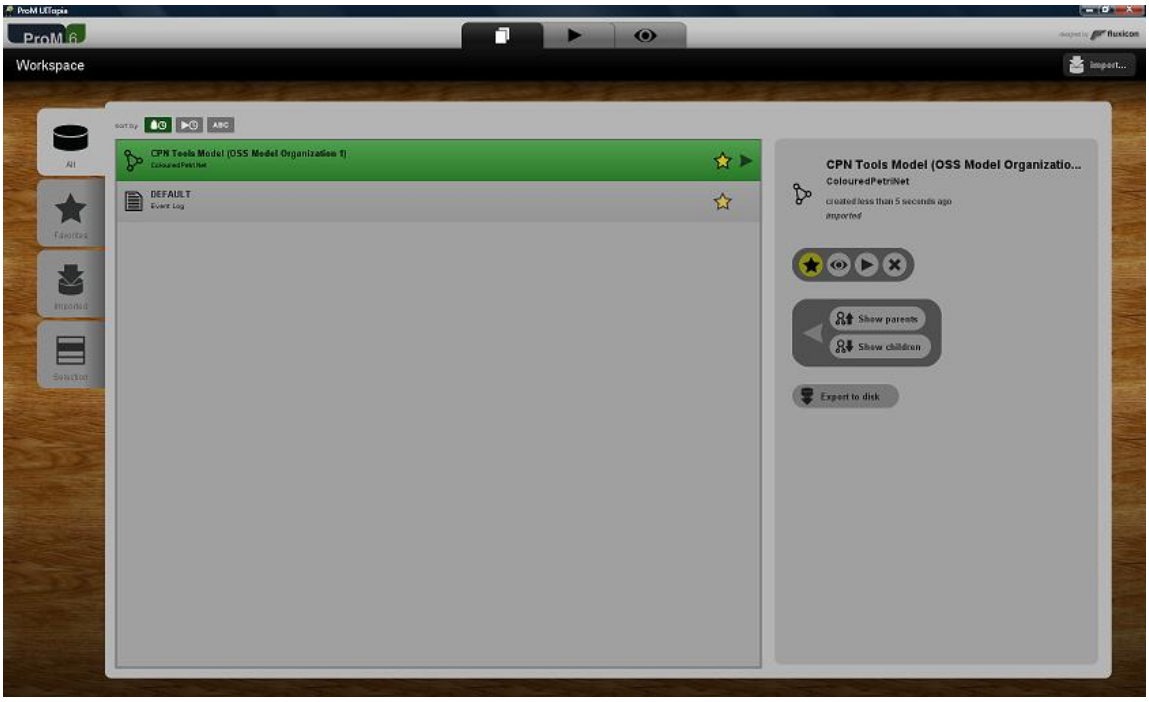

Figure 5.3: ProM 6 User Interface

The user interface of the core framework of version 6.1, the latest version of ProM, is shown in Figure 5.3.

ProM consists of multiple plug-ins, which can be seen as separate modules, each having their own functionality. The plug-ins are distributed as separate packages, such that users are able to only select those which are desired. Moreover, it allows developers to reuse specific code for their work.

For the creation of a collective transition system from multiple event logs, the TSMiner plug-in is used. This plug-in takes an event log as input and derives using various settings a transition system. Therefore, the implementation first retrieves all traces of all event logs and puts them in one log, the collective event log. Before constructing the collective transition system from this event log, decisions on which abstraction techniques to use have to be specified. Some of the most important abstraction mechanisms were already discussed in Section 3.2. Additional settings are further elaborated on in Section 6.1 of the next chapter.

After the collective transition system is created using the TSMiner plug-in, it is annotated with time information present in the collective event log using the TSAnalyzer plug-in of ProM.

Figure 5.4 shows the user interface of the Collective Transition System Miner. A resulting mined collective transition system is depicted in Figure 5.5.

The chosen abstraction settings influence the way the collective transition system has to be navigated. Therefore, the Multiple Log Recommender, which is described in Section 5.3, is implemented such that it is aware of these settings. A component which starts simulations using various parameters specifies which settings to use. This component is discussed in Section 5.4.

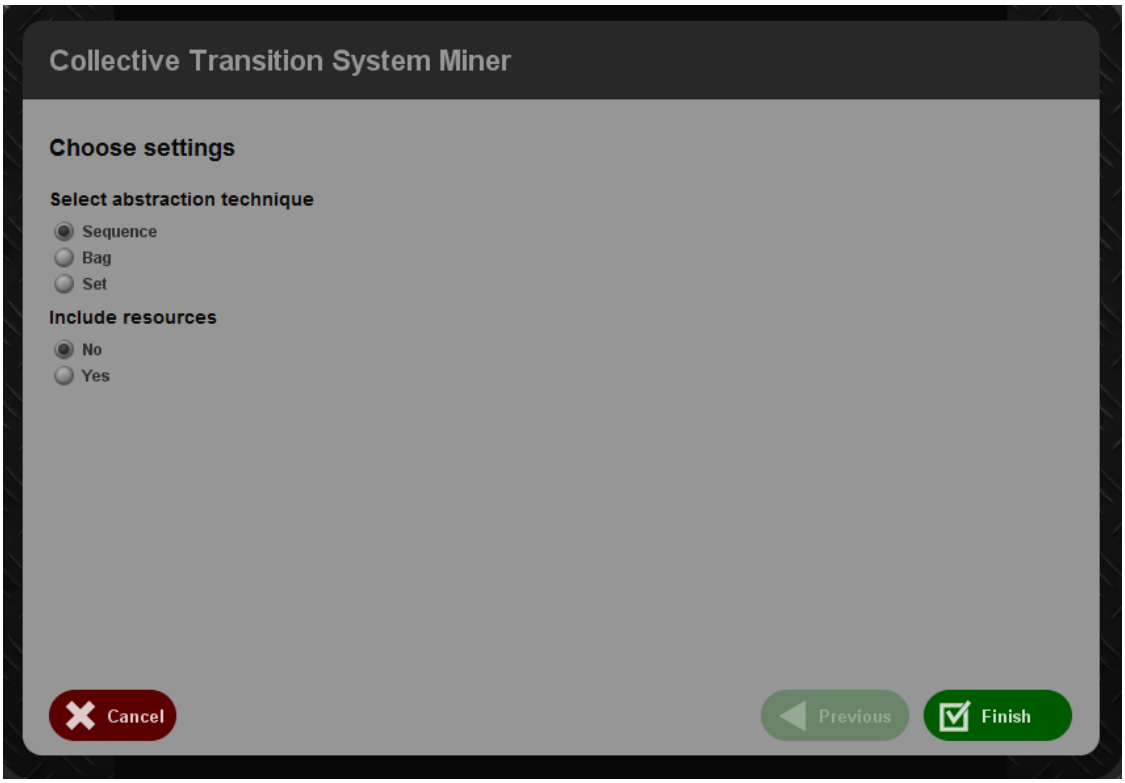

Figure 5.4: Collective Transition System Miner User Interface

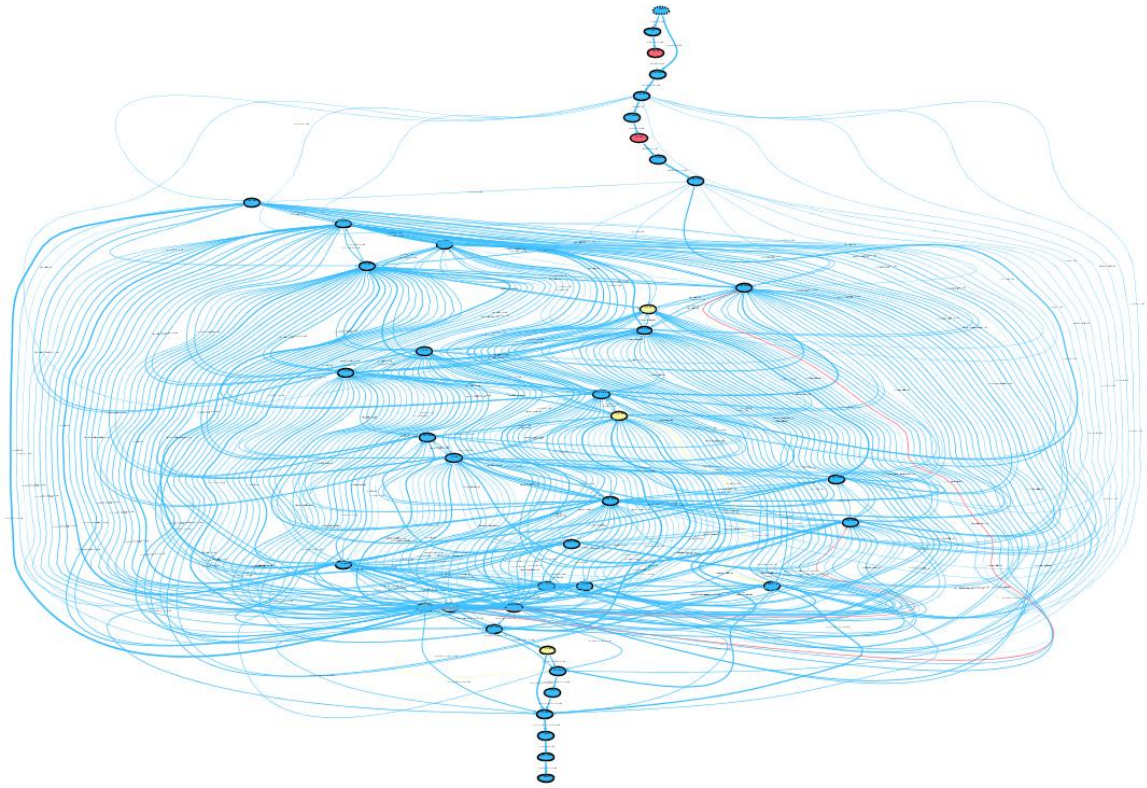

Figure 5.5: A collective transition system annotated with time information

5.2 Executable Process Models

This section describes the executable process models needed for this research. First, an executable process model from which an event log is generated is briefly discussed. Afterwards, a model which communicates with the Operational Support Framework of ProM and can request and process recommendations is explained. This implementation is inspired from [16], but some parts were changed or removed to make it suitable for this master project. The sub architecture is shown in Figure 5.6.

The executable process models are implemented using CPN Tools[2]. CPN Tools is a software tool for creating, analyzing and simulating Coloured Petri Nets [17], which were discussed in Section 2.2. By running the executable models in CPN Tools, business process executions of organizations can be simulated. Figure 5.7 shows the User Interface of CPN Tools.

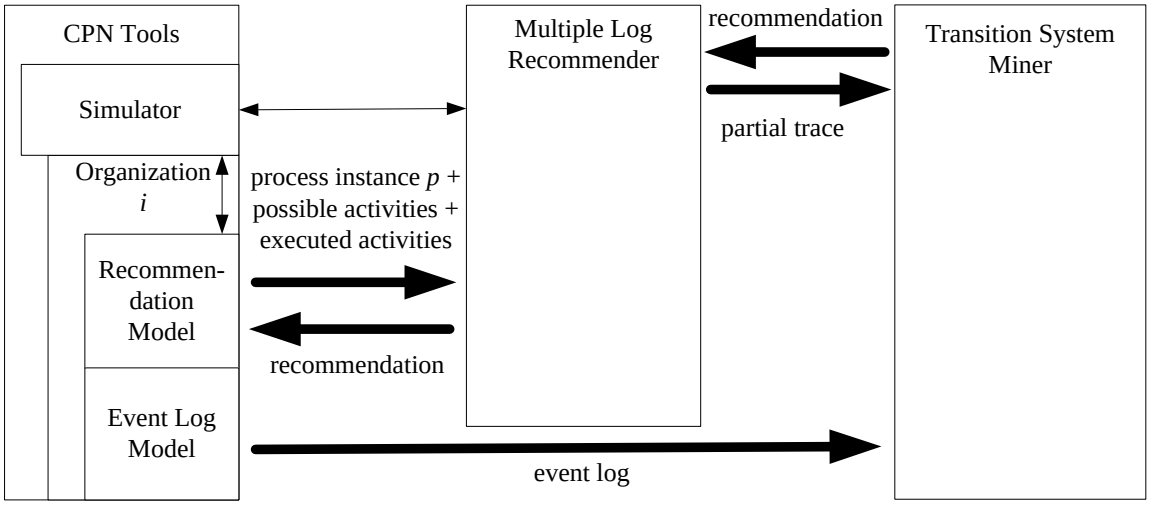

Figure 5.6: Architecture of the implementation highlighting Executable Process Models

To obtain a collective transition system of organizations 1...*n*, event logs 1...*n* are required. These event logs are retrieved by modelling the organizations using CPN Tools and simulating process executions. Recall from Section 2.3 that event logs have to be in either the MXML or XES format such that they can be used by ProM. For this reason, the external application ProM Import Framework[3] was developed. This is a framework that can extract MXML-formatted logs from all kinds of log data in other formats from any Process-Aware Information System [18]. The User Interface of this framework is shown in Figure 5.8.

To ensure that the event logs contain all relevant information, a library [6] is used which is referred to at every transition. In this way, the event log contains all required information as described in Section 2.3, such as case identifier, event name, event type, resource and timestamp. Because milliseconds are chosen as the time unit for the simulations in this work, the method in the library which calculates the timestamp is adapted.

[2] For more information, see http://cpntools.org/start
[3] For more information, see http://sourceforge.net/projects/promimport/

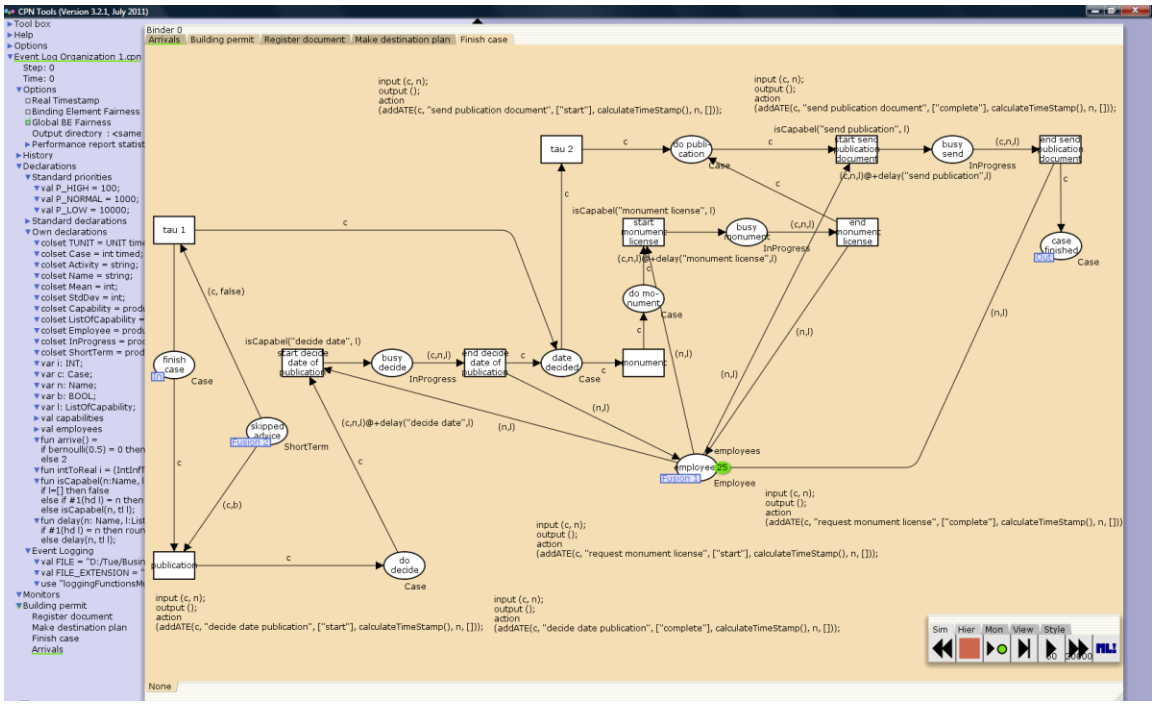

Figure 5.7: CPN Tools User Interface

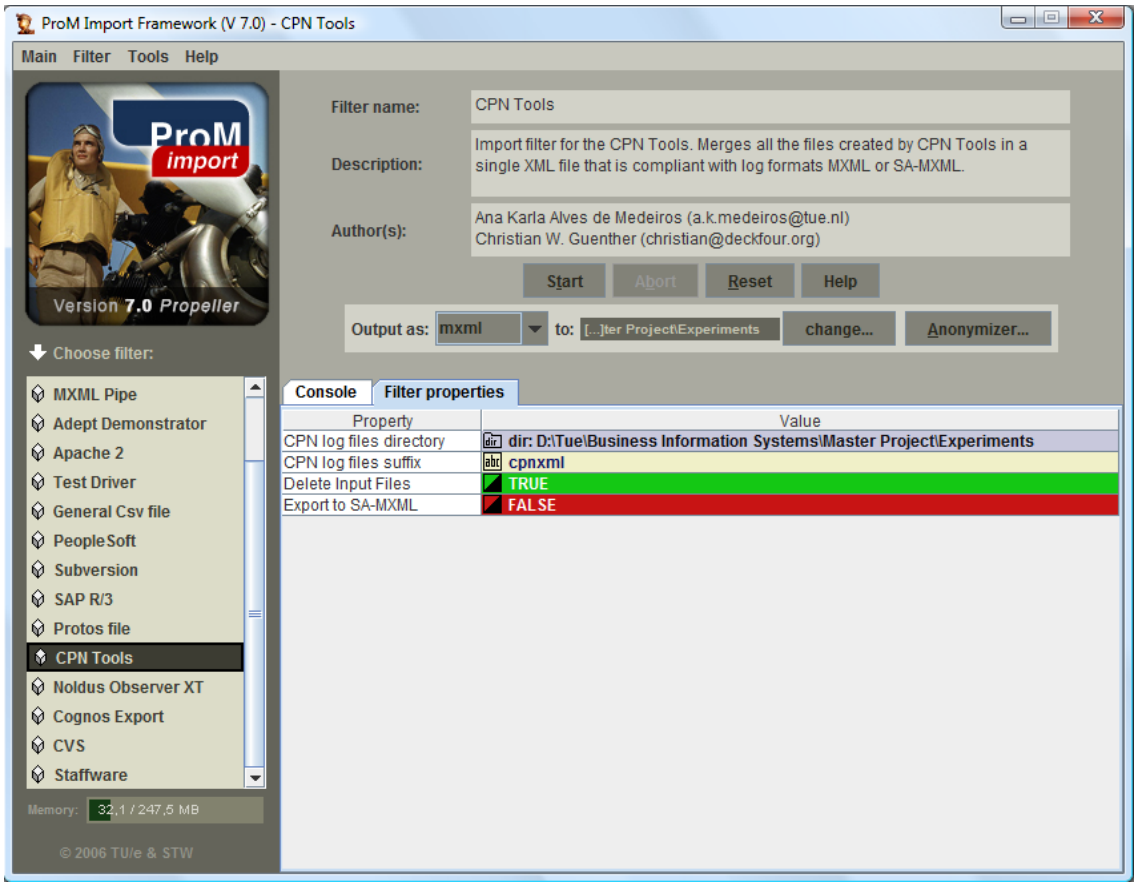

Figure 5.8: ProM Import Framework 7 User Interface

Now that event logs of organizations are created, a collective transition system from this historical data can be obtained. The Multiple Log Recommender navigates this process model for providing recommendations. To actually be able to both request and process these recommendations, executable models of organizations have to be created which perform these tasks. Furthermore, data of process executions of such organizations has to be analyzed in order to answer the research questions of this work.

To test the performance of organizations when recommendations based on the recorded behaviour of other organizations are followed, an executable process model reflecting an organization has to be constructed. Furthermore, this process model should be modelled such that for the states in the process where recommendations are requested, it is completely dependent of the provided responses by the Multiple Log Recommender. In other words, for those situations, the models themselves should not make any decision on the next activity that has to be carried out. Instead, the model should request recommendations to the Multiple Log Recommender about which activity to carry out next.

To achieve this, priorities are used in the Coloured Petri net. Values are assigned to transitions which imply the allowed order of firing: the transition which is enabled and has the lowest priority value can actually be executed. Using priorities, transitions related to carrying out work for a process instance have priority over transitions which are related with recommendation requests. In such a way, recommendations are only requested when no work for a process instance can be executed. This ensures that the execution of an activity by process instances which follow the recommendations corresponds to the execution of an activity by process instances that would make all decisions themselves.

In the Coloured Petri net implementation, three places are introduced to process recommendation requests and responses:

- **Offered** A place with for a each partial process instance a list of activities that currently can be executed
- **Selected** A place with activities that were returned by the recommendation service, at most one for a process instance
- **Completed** A place with (recommended) activities which were executed by a resource

Figures 5.9 and 5.10 depict a small example of the use of these three places for the execution of activities. The implementation in Figure 5.9 is responsible for the actual execution of all recommended activities, while the part of the coloured Petri net in Figure 5.10 takes care of the routing of the business process executions.

When activity ‘receive request’ is recommended by the Multiple Log Recommender, a token containing that activity, resource and the corresponding process instance identifier is created in places ‘Requested’ and ‘Selected’. At the same time unit *t*, transitions ‘Start Work’ (Figure 5.9) and ‘start receive request1’ (Figure 5.10) can fire.

The transition ‘Start Work’ models the execution of work for all recommended activities. A corresponding delay is added to the case and the event is added to the trace. When the current simulation time equals *t* + that delay, transition ‘Complete Work’ fires. This transition updates

the trace, sends the corresponding employee back to place ‘employees’ and creates a token in place ‘Completed’.

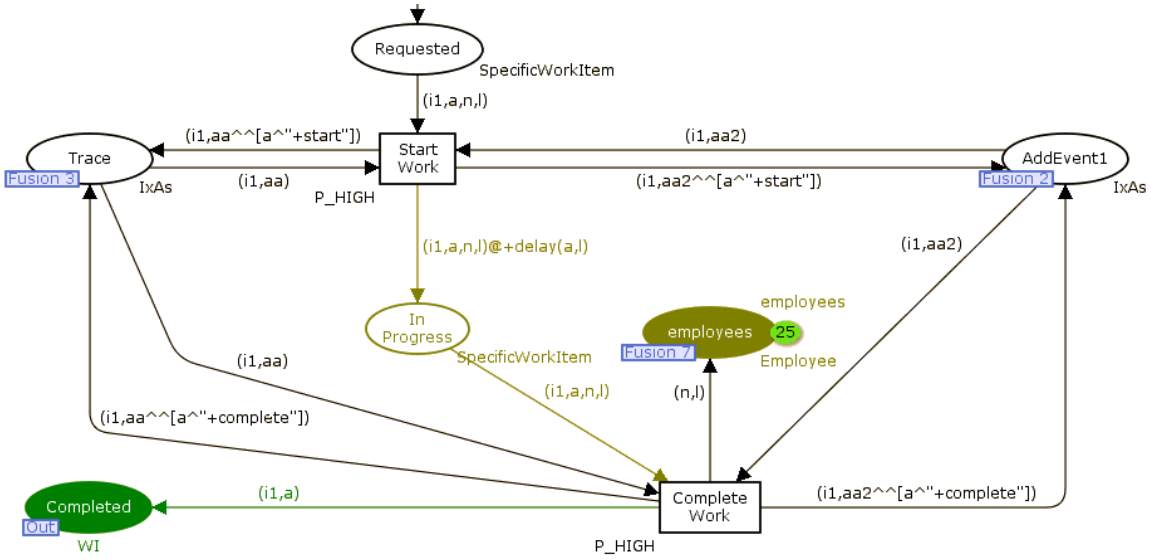

Figure 5.9: The actual execution of recommended activities

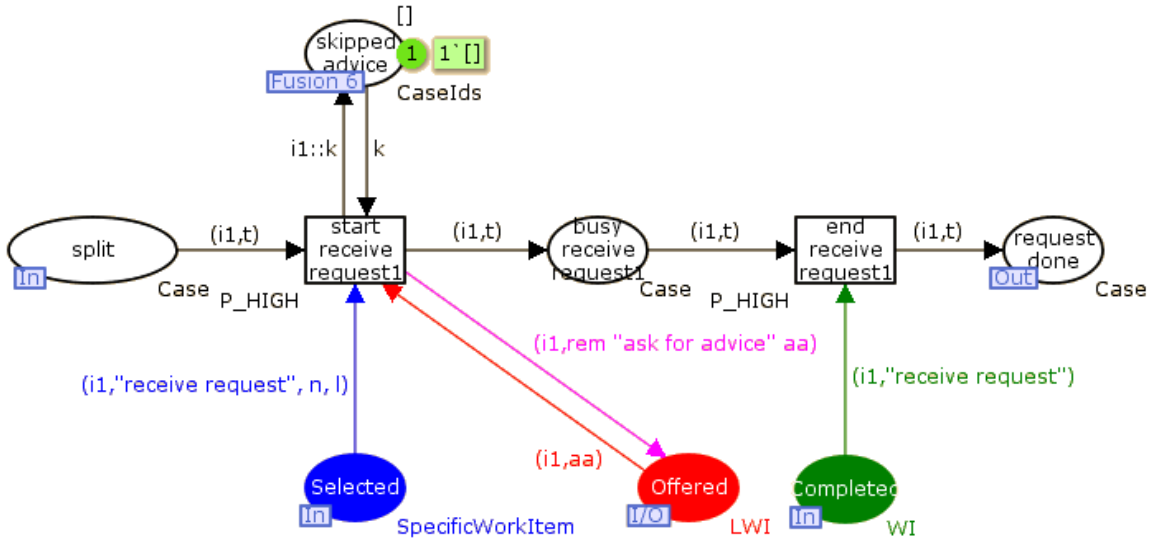

Figure 5.10: An example of the use of the places Offered, Selected and Completed

In Figure 5.10, the token in place ‘Selected’ enables the case in place ‘split’ to proceed to place ‘busy request’ by firing ‘start receive request 1’. Moreover, in the example, the activity ‘ask for advice’ is removed from place ‘Offered’. This means that this activity cannot be executed anymore for this process instance, since the Multiple Log Recommender will not be able to receive it as a possible activity. In such a way, an exclusive choice is modelled in an executable model which is not allowed to make this decision itself.

Recall that the moment the resource completed the activity ‘receive request’ by firing ‘Complete Work’, a token is placed in place ‘Completed’. This token contains the process instance and activity name and enables the model to fire transition ‘end receive request1’. A token is created in place ‘request done’ and the process execution can continue.

As illustrated by these examples, the ‘Selected’ and ‘Completed’ places are used to model the control-flow of the organizations. Both interface places contain information which allows process instances to proceed in the model and add new possible activities to the ‘Offered’ interface place. The next paragraphs elaborate on how these places are used in order to request and process recommendations respectively.

5.2.1 Request a Recommendation

When a process instance cannot proceed, because there is no unprocessed work in the interface places ‘Selected’ and ‘Completed’, a recommendation is requested about which activity in interface place ‘Offered’ to execute next. A request for a recommendation by a process instance of an organization contains the process instance identifier, which is a unique number for each process instance, a list of activities that were executed since the last recommendation request (if any), and a list of possible activities that can be executed next.

Such an approach solves most of the problems mentioned in Sections 3.4 and 4.3. Because the recommendation service has to recommend an activity which is present in the list of possible activities, it is not possible that a recommendation involves an activity that is not available or possible in the organization.

Figure 5.11 shows the start of the process of requesting a recommendation. For the case with identifier ‘1’, activities ‘ask for advice’ and ‘receive request’ can be executed. The transition ‘Select Instance’ will first fire and select all activities from ‘Offered’ for which a resource is available that has the capabilities to execute it, along with that resource. Then, transition ‘Perform Query’ sends a recommendation request for this process instance, together with these activities and a list of executed activities since the last recommendation request for that case (if any) in place ‘AddEvent1’. It does so by creating a token with this information in place ‘Recommend’.

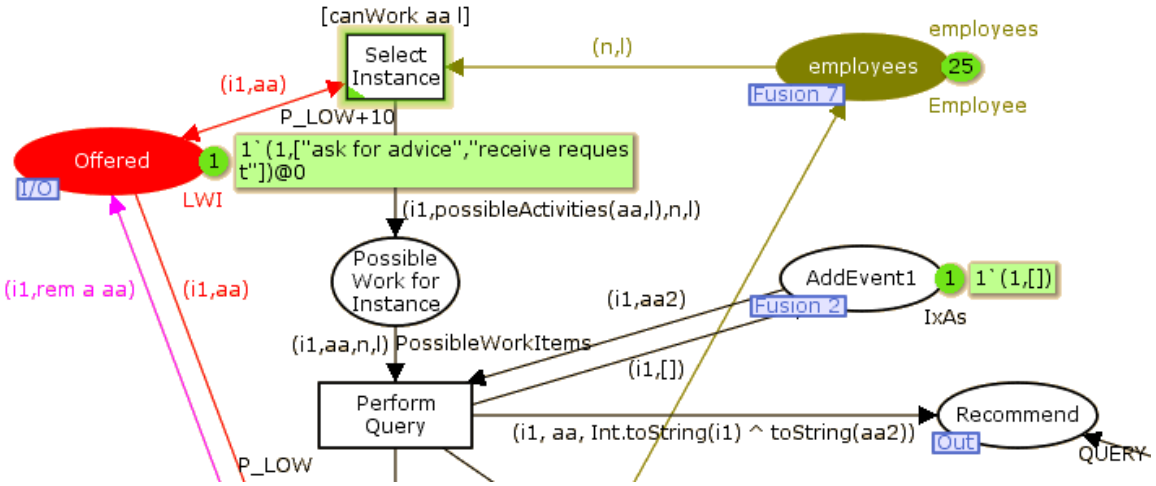

Figure 5.11: The process of requesting a recommendation

The requests for (and responses of) a recommendation are handled via a subpage ‘OSS Service’. Although in the Coloured Petri net model itself no interesting functionality takes place there, this subpage is replaced by the ‘Multiple Log Recommender’, which receives the requests and sends the responses. Section 5.3 elaborates on this in more detail.

5.2.2 Process a Recommendation

Note that while a recommendation request is send to the Multiple Log Recommender and no response is yet received, the executable model should wait without performing any tasks. The reason for this is that business process executions have to be simulated such that possible waiting times have no effect on the performance of executions. Thus, the actions performed to reply a request with a recommendation should have no influence on a process instance.

However, using this implementation, the decision on what to do next is not atomic. Even worse, the executable model is not aware of the fact that a response will be received by the Multiple Log Recommender. This could lead to a situation in which CPN Tools increases the clock time to the lowest time a transition in the model is enabled. To solve this issue, a transition ‘step while waiting’ is included in the CPN model with the highest priority. This transition fires if and only if a recommendation request is send but not yet replied.

When a response is returned, the Multiple Log Recommender places a token in place ‘Response’. This token contains the process instance identifier and the recommended activity. The selected resource now has to carry out this activity as soon as possible and the chosen activity is removed from the list of possible activities in place ‘Offered’. Recall that the model makes sure that only one recommendation can be requested at a time. Figure 5.12 shows an example in which the activity ‘receive request’ is recommended for process instance ‘1’.

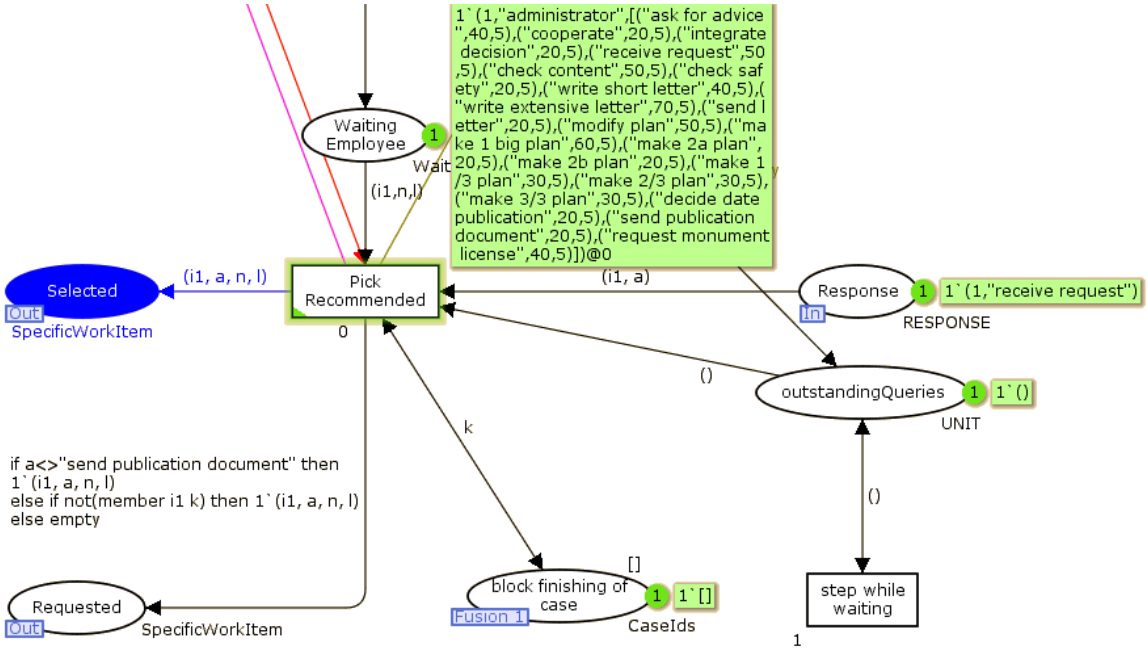

Figure 5.12: The process of handling a recommendation

When for a process instance all (recommended) activities have been carried out, it is finished. The starting time of the process instance is then subtracted from the current time to calculate the total flow time for that case. In this way, minimum, maximum and average flow times of cases are computed for each experiment, which is discussed in Section 5.4.

Finally, the executable model contains a construct that is used for the end of the entire simulation of the business process. New cases arrive in the business process following a time distribution and are considered as 'pending'. The above described finishing of a process instance removes it from the pending list. When the set simulation length ends, and all cases are finished, a final transition fires which places a token in place 'Recommend' of Figure 5.11 with the value '(0,["END OF SIMULATION"],"END OF SIMULATION")'. As will be explained in Section 5.4, the Multiple Log Recommender is then able to interpret this data and stop the simulation process.

5.3 Multiple Log Recommender

The previous section discussed the implementation of an executable model which requests and processes recommendations. This section describes the developed application that receives these requests and provides a reply in terms of a suggested activity. This application is called 'Multiple Log Recommender' and is implemented as a ProM plug-in. It makes use of Operational Support Service 2.0 in order to communicate with the Coloured Petri net implementations described in the previous section. These implementations are embedded in an executable model which represents a business process of an organization. Figure 5.13 visualizes the architecture of the Multiple Log Recommender.

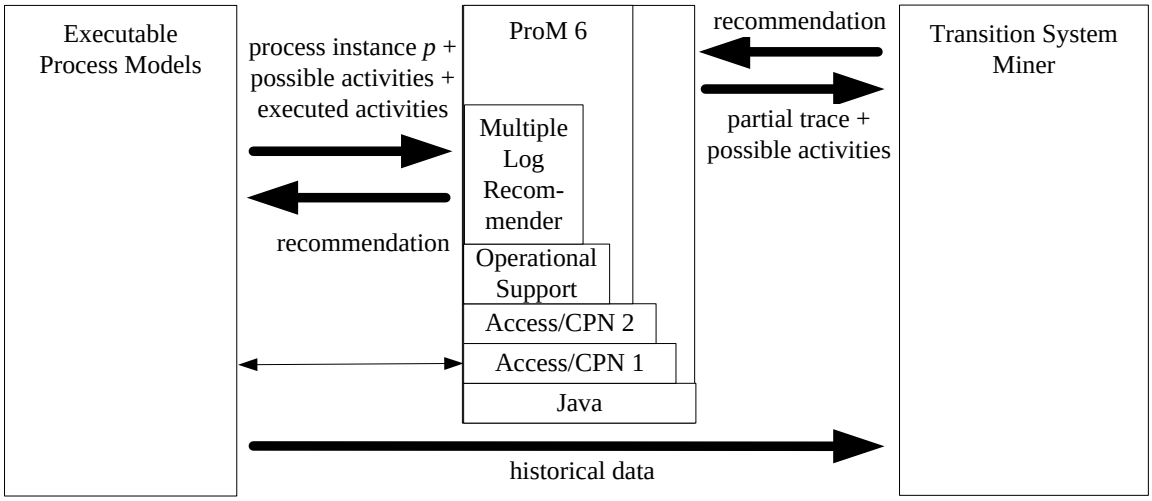

Figure 5.13: Architecture of the implementation highlighting Multiple Log Recommender

The Multiple Log Recommender takes three collective transition systems, together with their corresponding time annotations, as input. These process models were elaborated on in Section 5.1. Moreover, a percentage which expresses the fraction of times a recommendation is

addressed has to be specified. This number is introduced to simulate situations in which users do not follow a recommendation, but choose to carry out another activity instead. Because in the implementation, the model always carries out the responded activity, this activity itself is deviated from a 'normal' recommendation. Thus, the probability of this is indicated by this number. For instance, 0 implies only random behaviour (and thus represents simply picking an arbitrary activity to execute all the time).

The recommender makes use of three collective transition systems, derived using the abstraction techniques sequence, bag and set respectively. The reason for this is twofold: first of all, when offering a recommendation, a level of certainty can be expressed. A suggested activity derived from the collective transition system mined using the sequence collection type has a higher level of certainty than recommendations resulting from the transition systems obtained by applying the bag and set abstraction techniques. Secondly, when partial process instances cannot be mapped onto the transition system constructed using the sequence abstraction, instead of having to return a random recommendation, it can still be possible to provide a recommendation using the collective transition system of bag (and possibly one abstraction level higher, the transition system of set). Thus, through this approach, the overall quality of recommended activities will increase.

As shown in paragraph 5.2.1, a recommendation request contains a process instance identifier, a list of possible activities and a list of executed activities since the last recommendation request (if any). Additionally, the Multiple Log Recommender makes use of four variables for each process instance for which recommendation requests are received. These variables are needed in order to provide recommendations for as much situations as possible and are stored in a session object to make sure that each recommendation request of a particular process instance has access to them. For each process instance, a separate session object is created. The four variables are listed below:

- **trace** — A String that contains all the activities that were executed for the process instance
- **current abstraction** — A String that indicates the current abstraction level for the process instance (sequence, bag or set)
- **current position candidates** — A List that contains the possible current states (if any) in the collective transition system of the current abstraction
- **lost state** — A String that indicates whether the state was lost during the previous recommendation request of the process instance (if any)

For each recommendation request, the executed activities since the previous request (if any) are added to the trace of the process instance. This trace is needed in scenarios where the state of a process instance in a transition system is lost or when a recommendation cannot be provided and a higher level of abstraction is needed. Using the trace, the current position candidates can be retrieved using a higher level of abstraction, such that a recommendation better than random can possibly still be provided. Moreover, the state is lost when no states in a collective transition system can be mapped on the partial trace of a process instance. Finally, the

current abstraction variable indicates which transition system is used in order to provide a recommendation. When a recommendation fails, for whatever reason, the value is changed from sequence to bag or from bag to set.

The next paragraph describes the different scenarios that can occur while trying to provide a recommendation.

5.3.1 Scenario's while Providing a Recommendation

There are multiple causes why there are a lot of scenarios which can occur while trying to determine a recommendation. These include the loss of a current state at any moment in the process, no possible activities which can be carried out from any of the current position candidates, etc. This paragraph explains the approach taken to overcome these issues and to provide a recommendation with as much certainty as possible.

First, depending on the percentage of recommendations, a chance of providing a 'normal' or random recommendation is computed. When a random recommendation is chosen, one of the possible activities is randomly picked. Otherwise, the best alternative among the possible activities is chosen.

In order to choose the best option, first the states that can be the current state of the process instance have to be determined. Depending on the collective transition system and current abstraction technique, there can be multiple of such states. These states are called current position candidates. The procedure to determine these candidates is described in the next paragraph. The current position candidates of the process instance in the current collective transition system are taken as input if already available; else the initial states from that transition system are taken. Then, the possible activity is recommended which equals the outgoing transition that connects any of the current position candidates with the state that has the lowest average remaining time from all the reachable states. Additionally, the current position candidates are stored in the corresponding session object.

As mentioned before, when one of the intended operations above does not succeed, alternative steps have to be taken in order to still be able to provide a recommendation. These scenarios are included in the flowchart presented in Figure 5.14. It shows the steps which need to be taken before a recommendation can be provided. Then, the procedure continues as visualized in Figure 5.15.

The next paragraphs describe the implementations of the methods to determine the current position candidates and to generate a recommendation in more detail.

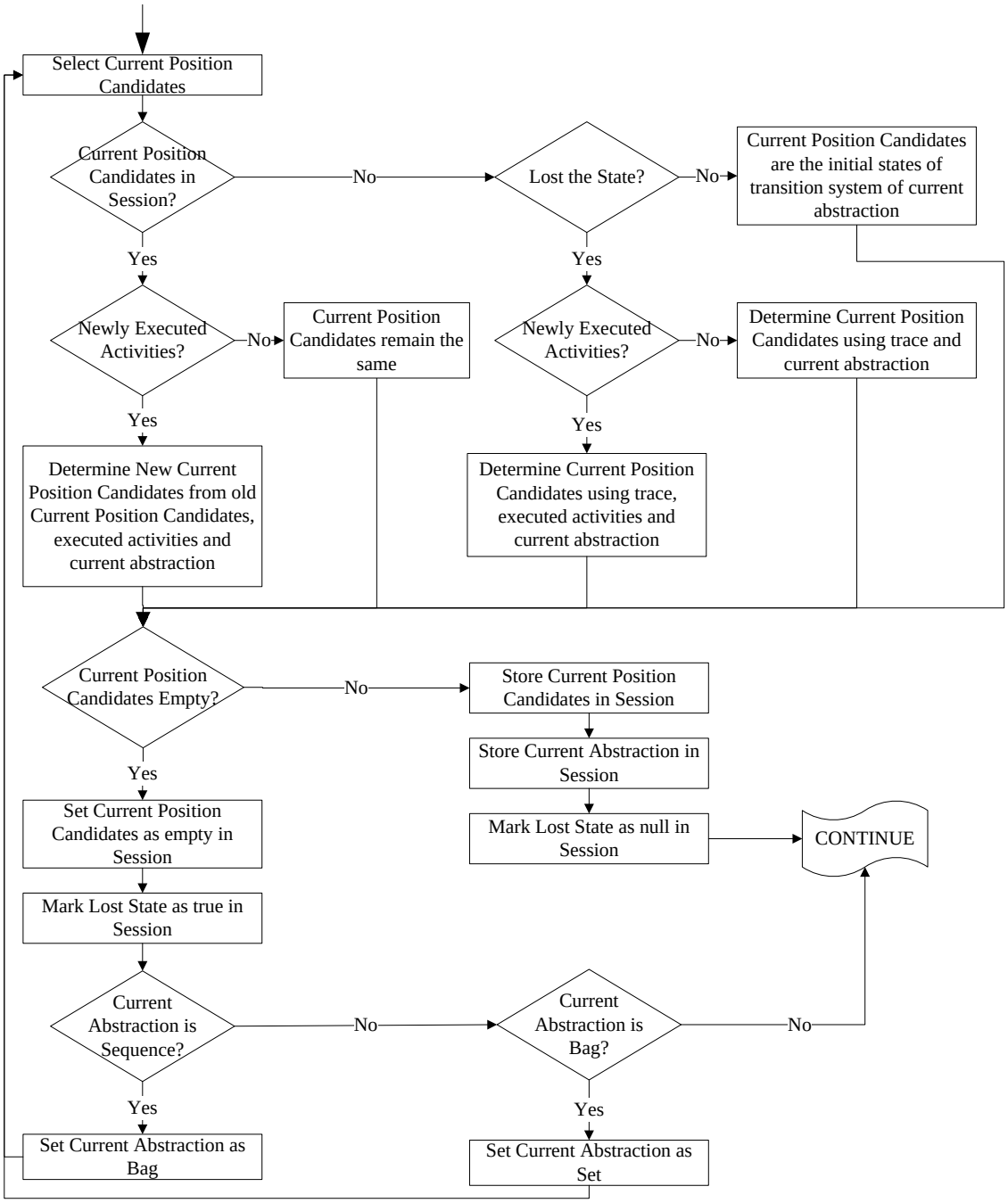

Figure 5.14: A graphical depiction of the steps that need to be taken for a recommendation

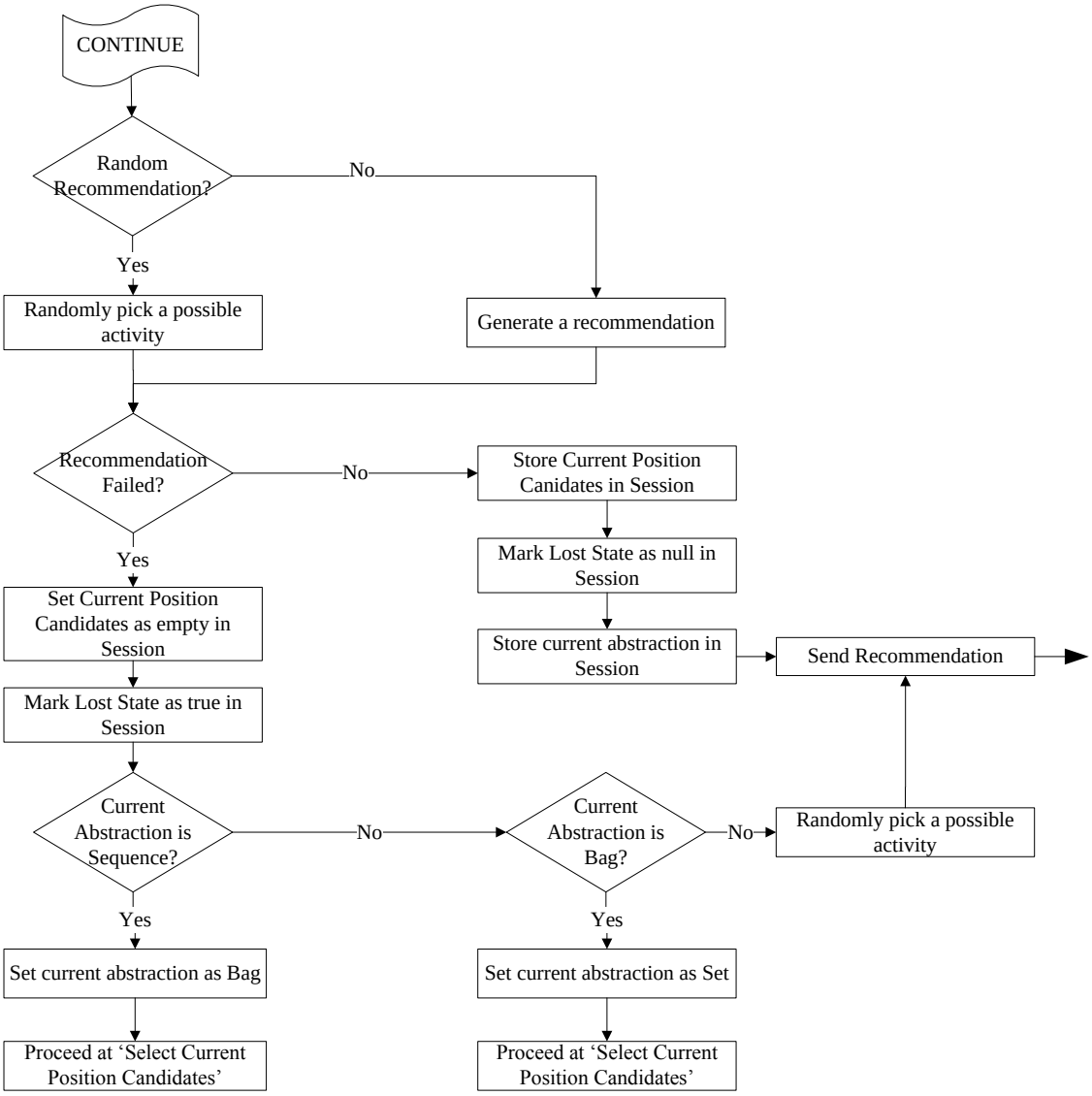

Figure 5.15: Additional required steps when generating a recommendation

5.3.2 Determining Current Position Candidates

This paragraph describes the algorithm that is used to determine the current position candidates of a process instance. Recall that the current position candidates are the possible states in which a process instance can be situated.

In the scenario in which no activities were executed since the last recommendation request for a process instance, the current position candidates, which already were stored, are simply used. If no recommendation was requested before, the current position candidates are the initial states of the transition system with the sequence abstraction.

However, when activities were executed since the last recommendation request, independent of the fact whether the current position candidates were stored or is a set of initial states of the transition system, these executed activities have to be replayed on the collective

transition system to determine the current position candidates. The procedure that has to be followed for this purpose depends on the abstraction technique that was used to construct the collective transition system. The algorithm for the sequence abstraction technique is presented in pseudo code in Listing 5.1; the pseudo code of the multi-set (bag) and set implementations is given in Listing 5.2.

```
if initial_states is empty then
   initial_states ← initial states of transition system of current abstraction
end
foreach state s in initial_states do
   still_to_replay_activities ← executed activities
   still_candidate = true
   current_state = s
   while not(still_to_replay_activities empty) and (still_candidate is true) do
      next_state_found = false
      next_activity ← first activity of executed activities
      states possibilities_to_go = all states connected from current state
      if possibilities_to_go is empty then
         still_candidate = false
      end
      foreach state t in possibilities_to_go do
         if (transition connecting s and t) equals next_activity then
            current_state = t
            next_state_found = true
            remove next activity from still_to_replay_activities
            if still_to_replay_activities is empty then
                 if current_state has no outgoing transitions then
                    current_state is a final state
                 else
                    add current_state to current position candidates
                 end
             end
         end
      end
      if next_state_found is false then
         still_candidate = false
      end
   end
end
return current position candidates
```

Listing 5.1: Current position candidates sequence algorithm

Note that the algorithm in Listing 5.1 is started with the previously stored current position candidates as initial states. If no current position candidates were stored before, this list is empty and the initial states of the transition system of the current abstraction are used. Then, the activities which were executed since the last recommendation are processed in a sequential way, preserving both the order and multiplicity of these activities. Until all executed activities are replayed, the collective transition system is explored for all initial states in the search for

transitions which equal these activities and connect the current state with a certain other state. The states that are reached when all executed activities are replayed are returned as current position candidates.

The following algorithm navigates the collective transition systems derived using a bag or set abstraction technique when the current abstraction is bag or set respectively.

```
if initial_states is empty then
   initial_states ← initial states of transition system of current abstraction
end
foreach state s in initial_states do
   still_to_replay_activities ← executed activities
   store pair of s and still_to_replay_activities in alternative_paths
end
while alternative_paths is not empty do
   foreach state s in alternative_paths do
      states possibilities_to_go = all states connected from s
      foreach state t in possibilities_to_go do
         foreach activity a in still_to_replay_activities of s do
            if (transition connecting s and t) equals a then
                remove a from still_to_replay_activities of s
               if still_to_replay_activities is empty then
                  add t to current position candidates
               else
                  store pair of t and still_to_replay_activities in
                  alternative_paths
               end
             end
         end
      end
      remove pair of s with still_to_replay_activities from alternative paths
      proceed with other state s in alternative_paths
   end
end
return current position candidates
```

Listing 5.2: Current position candidates multi-set/set algorithm

The algorithm shows that all states, reachable from a current position candidate with a transition that equals an activity which still has to be replayed, are taken as a path. Every combination of a state and list of activities to replay is stored. When a state is reached and all activities have been replayed, that state is added to the current position candidates.

Note that the algorithm is the same for both the multi-set (=bag) and set abstraction techniques, but that for the set abstraction technique, the executed activities list is preprocessed first such that possible multiple occurrences of the same activity are removed from it.
Now that the candidates for the current states in which the process instance can be situated are retrieved, the recommendation itself can be determined. This is explained in the next paragraph.

5.3.3 Determining a Recommendation

Depending on the percentage chance that a recommendation has to be followed, either a recommendation is requested or an activity of the list of possible activities is randomly chosen. When a recommendation is requested and the current position candidates for the process instance in consideration are determined using one of the algorithms described in the preceding paragraph, the following procedure is carried out:

```
foreach state c in current position candidates do
   states possibilities_to_go = all states connected from c
   foreach state s in possibilities_to_go do
      keep_possibility = false
      foreach activity p in possible_activities do
         if transition connecting c and s equals p then
            keep_possibility = true
         end
      end
      if keep_possibility is false then
         remove s from possibilities_to_go
      end
   end
   if not(possibilities_to_go is empty) then
      current_best_state_to_go = state s from possibilities_to_go with lowest
                                 average remaining time
      if best_state_to_go is null then
         best_state_to_go = current_best_state_to_go
         best_current_state = c
      else if average remaining time of current_best_state_to_go < average
           remaining time of best_state_to_go then
          best_state_to_go = current_best_state_to_go
          best_current_state = c
      end
   end
end
return transition connecting best_current_state and best_state_to_go
```

Listing 5.3: Recommended activity algorithm

The pseudo code in Listing 5.3 works as follows: for any current position candidate, all states that can be reached with a transition that is equally labelled as a possible activity are navigated. Recall from Section 3.4 that average process times are used for determining a recommendation. Therefore, from these states, the state with the lowest average remaining time is selected. If this state has a lower average remaining time than all earlier retrieved ‘best’ states of other current position candidates, it is chosen as best state to go. Finally, the activity that corresponds to the transition connecting the current position candidate with this best state is recommended.

Furthermore, after carrying out this algorithm, the current position candidates of the process instance are stored in the corresponding session object. Moreover, the executed activities

are removed from the session object and placed in the trace variable, since this list of activities has already been replayed on the collective transition system when determining the current position candidates.

The next section discusses the implementation which serves as an overarching application that invokes all components previously described in this chapter for simulations with different parameters.

5.4 Experiment Plug-in

The prior sections described the various components of the implementation of this work. To answer the research question of Chapter 1, experiments with a varying set of parameters have to be carried out. These experiments are elaborated on in the next chapter. This section explains the implementation of a ProM plug-in that invokes different instantiations of collective transition systems, executable models and the Multiple Log Recommender for this purpose.

As the architecture in Figure 5.13 indicates, the Multiple Log Recommender makes use of Operational Support Service 2.0. This is an implementation of the framework described in Section 3.1 in ProM. The framework provides an infrastructure for communication between the process instances in the executable model and the Multiple Log Recommender. These process instances are implemented as threads, which can be seen as independent executions of a program. For each process instance, an input and output channel is used. The input channel is connected to the place ‘Recommend’ as discussed in Paragraph 5.2.1. The output channel is connected to the place ‘Response’ as discussed Paragraph 5.2.2.

Because ProM has no information whether the simulation in CPN Tools is busy or finished, Paragraph 5.2.2 showed a workaround to solve this problem. The receipt of a recommendation request with that specific data is implemented as a sign to the plug-in to close the channels of a simulation and start a new experiment. Unfortunately, the Operational Support Framework was designed such that it is not guaranteed that a simulation always finishes correctly. The reason for this problem is that threads are not managed properly and therefore the execution can get stuck or raise errors. As a consequence, the simulations are unreliable in the sense that there always is a chance the experiment plug-in stops. Parameters then have to be changed manually to proceed with remaining experiments. As a result, for this research the execution of simulations is relatively time-expensive.

When a simulation finished correctly, data is stored before a new experiment is started. From this data, the following statistical information is computed.

- **Nr. of process instances** The number of process instances for which at least one recommendation was requested
- **Nr. of recommendations** The total number of recommendations
- **Nr. of sequence recommendations** The number of recommendations which were provided with the highest level of certainty (sequence)
- **Nr. of bag recommendations** The number of recommendations provided by making use of the collective transition system which was created using the bag abstraction

- **Nr. of set recommendations** The number of recommendations using the set abstraction technique
- **Nr. of random recommendations** The number of random recommendations, caused by representing user behaviour of not following recommendations, and partial traces which could not be mapped onto any collective transition system

Additionally, all data regarding the behaviour of process instances of an organization is recorded. This enables computation of various performance related numbers:

- **Total process time** The total process time of all process instances
- **Largest process time** The process time of the worst performing process instance
- **Shortest process time** The process time of the best performing process instance
- **Best trace** The corresponding trace of the best performing process instance
- **Average process time** The average process time of a process instance

The statistics and performance related numbers presented above are derived for each simulation. This implies that these numbers can be compared with their counterparts of simulations using different parameters. Thus, by making use of results of experiments using varying historical data, organizations and other relevant settings, the research questions of Section 1.2 can be answered.

The next chapter describes an organization from which similar variants are constructed. From these organizations, executable models and event logs are derived that are used for experiments and computation of the numbers above.

Chapter 6

Experiments

Section 6.1 first describes the approach used to obtain a realistic and as representative as possible set of experiments by reducing the problematic, initial experiment size and its corresponding problems. Then, Section 6.2 explains a configurable process model which represents a business process of an organization that contains relevant control-flow configuration patterns discussed in Section 4.2. This model is used to obtain 4 variants that represent similar organizations. The deviations in terms of control-flow patterns are indicated in Section 6.3. Finally, the results from the experiments using the implementation of the previous chapter for the 5 similar organizations are listed and elaborated on in Section 6.4.

6.1 Experimental Setup

In order to carry out the experiments, a ProM plug-in was discussed in Section 5.4 that automatically loads the relevant process models and event logs. Moreover, it starts the Operational Support Service 2.0, with which the Multiple Log Recommender communicates. Also, the communication between the process model in the PAIS and the Multiple Log Recommender is setup by creating mappings between the Operational Support Service and the interface places, which are embedded in the executable process models that represent an organization. Furthermore, it creates the three collective transition systems; one for each of the abstraction techniques sequence, bag and set. Together with a number, which represents the intended percentage of recommendations, they are given as input for the Multiple Log Recommender, which was described in Section 5.3.

While running the experiments, the plug-in keeps track of the carried out activities and associated timestamps, such that after an experiment the flow times of process instances in an experiment can be computed. Moreover, the plug-in maintains and stores various statistics, such as the number of process instances and the number of recommendations with a corresponding certainty. These statistics were explained in Section 5.4.

For the experiments, multiple parameters play a role. Next to the process dimension containing various control-flow configuration patterns, different settings for 4 other dimensions can be used. First of all, resource settings are incorporated within the models. Second, the collective transition systems which are used to base the recommendations on can be mined with

different settings. Moreover, they can originate from any combination of historical data of the other organizations. Finally, also the percentage of recommendations that are actually followed by users can vary per experiment. Figure 6.1 visualizes the resulting potential size of experiments, which is the product of the settings of each parameter listed above.

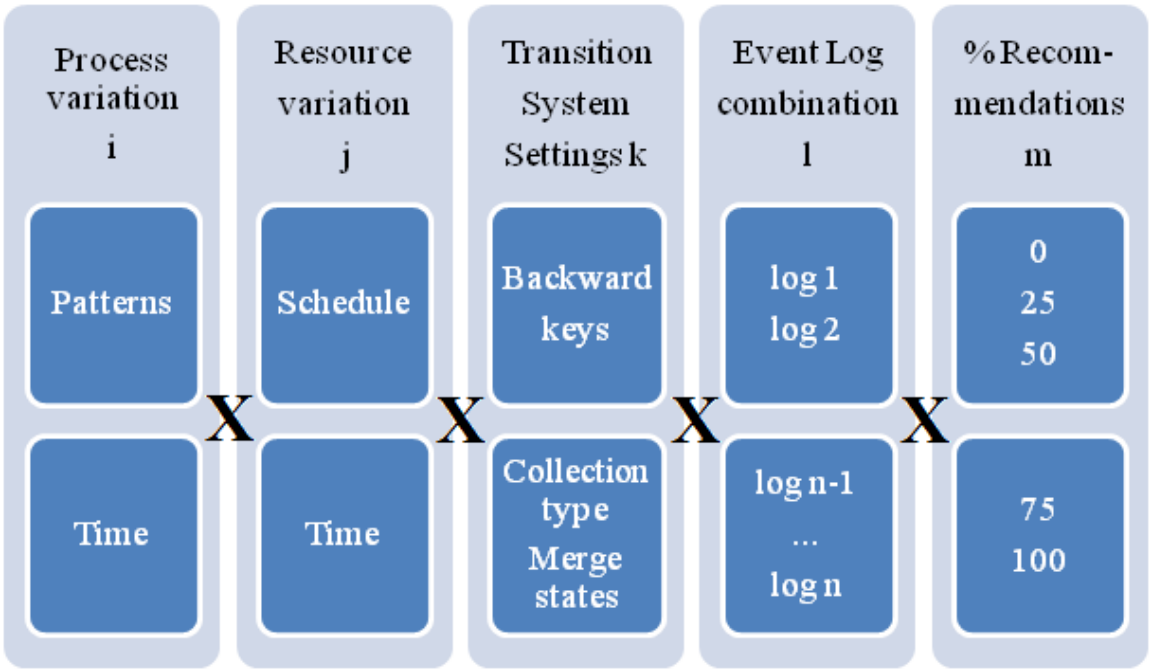

Figure 6.1: Potential size of experiments

In Chapter 4, 9 different control-flow patterns were identified. Moreover, for each pattern a number of configurations can be set. A process variation *i* can contain any combination of such settings for any combination of control-flow patterns. Therefore, the number of possible process variations is extremely large.

For the resource dimension, different number and types of resources can be selected that carry out the activities contained in the control-flow patterns of the process model. Their level of expertise, competences and schedules can vary. Again, any combination of these possibilities can be chosen as the resource variation *j*, which yields a very large number.

The transition system settings dimension contains a much smaller set of combinations. However, the size of the experiments is still increased with ($i * j * l * m$) experiments for each transition system setting *k*.

Additionally, each combination of historical data of the resulting process models of combinations $i * j$ can be chosen as the data to mine the collective transition systems from. This leads to $(2^{(i*j)}-1)$ possible combinations for the event log dimension.

Finally, 101 settings for the recommendation dimension are available, since each percentage from 0 – 100 is a possibility.

From these computations, it is clear that the number of possible experiments is extremely big and grows drastically with every added choice in parameters. It therefore is not reasonable, and maybe even impossible regarding time, to conduct every single experiment. For this reason,

choices are made such that the set of experiments is realistic, and still all control-flow patterns and configuration settings are covered.

To bring the number of experiments back to a feasible number regarding facilities and time, for the process dimension, it is decided to choose 5 different process models. Therefore, 4 organizations are derived from a configurable process model that is elaborated on in the next section. The following paragraphs contain actions that are taken to reduce the number of variations for the other dimensions, starting with the transition system dimension.

6.1.1 Transition System Dimension

The decision to make use of three transition systems, one for the sequence, bag and set abstraction technique respectively, resulted in two advantages. It enables the expression of a level of certainty for a recommendation. Secondly, recommendations can be provided in more scenarios. Additionally, it also provides an advantage towards the number of experiments required. This number is reduced with 2/3; instead of having separate experiments for sequence, bag and set settings, one experiment is carried out having all three of these collection types.

However, still 8 possible combinations of relevant transition system settings remain. To reduce this number to 2 settings for the transition system dimension, it is decided not to make use of classifiers related to forward keys. Forward keys are concerned with events that will occur. Since the recommendation service would base its decision on what activity to recommend on the activities that already have occurred, it does not make sense to separate states of a transition system based on the future. By similar reasoning, the transition system setting ‘Merge states with identical outflow’ is not used for the experiments. As a consequence, the only 2 transition system settings that are used are:

- Backward keys: MXML Legacy Classifier, Event Name
- Backward keys: MXML Legacy Classifier, Event Name and Merge states with identical inflow

6.1.2 Resource Dimension

Because the goal of these experiments is to test how good or bad recommendations based on the recorded behaviour of other organizations are, and furthermore to discover which control-flow configuration patterns have a positive or negative impact, it is important to keep other variables as stable as possible. Otherwise, a change in these variables might be the reason for varying or divergent results, and possibly wrong conclusions are drawn. Therefore, the settings for the resources, such as number of resources, schedules, competences and level of expertise will remain constant across experiments. Thus, for the resource dimension there is only 1 setting.

6.1.3 Event Log Dimension

Recall that in order to assess whether organizations can learn from other organizations, the performance has to be measured and compared. So, for the event logs dimension, first an experiment has to be carried out with an organization's own recorded behaviour only. Then, another experiment has to be executed, but this time with all historical data, except from the organization itself. This reduces the amount of experiments with $(2^n-1)-1$, with n the number of event logs. However, still it can be discovered whether improvement is possible by the use of cross-organizational data.

For each organization, or configurable process model variation, experiments with 5 different recommendations percentages (0, 25, 50, 75 and 100) are carried out. This reduces the number of settings for the percentage of recommendations dimension from 101 to 5.

6.2 General Process Model

To obtain 5 similar business processes, an organization carrying out a business process is modelled taking the role of a configurable process model. Recall from Section 4.1 that configurable process models integrate the different variants of a business process into a single model. In other words, from a configurable process model different models can be constructed that carry out a business process in a varying way. Then, these different models incorporate the functionality described in Section 5.2 to obtain executable models that create event logs of the organization and request and process recommendations respectively.

Because this master project was executed in the context of the CoSeLoG project, a model depicting a simplified process within a municipality, namely handling the request for a building permit, is chosen. Since no suitable process model of such a business process was available, the model that is used is inspired by a real-life process. It is used as a running example in the remainder of this thesis.

Furthermore, the configurable process model is chosen such that it contains relevant control-flow configuration patterns which were given in Section 4.2. The overview of this building permit process model is shown in Figure 6.2. The process model is described in the next paragraph. Paragraph 6.2.2 elaborates on the control-flow configuration patterns in the model.

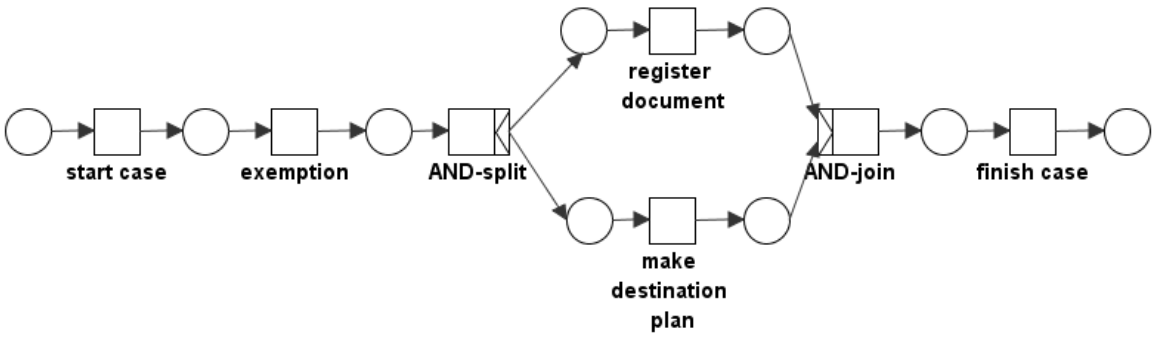

Figure 6.2: Overview of the configurable process model of handling building permits

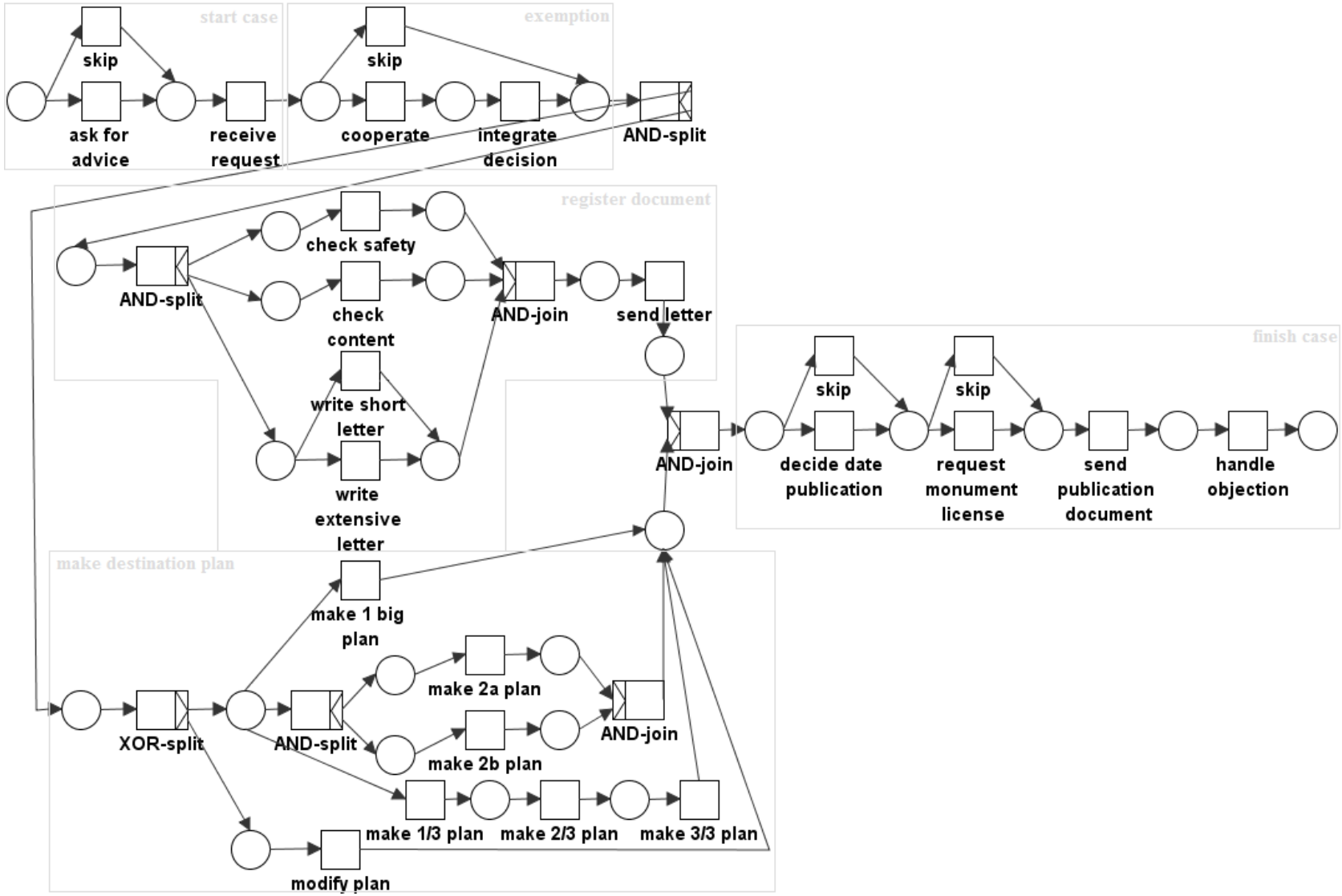

Figure 6.3: Overview of the configurable process model without hierarchy

6.2.1 Description of the Model

Figure 6.3 shows the business process without hierarchy. The start of the process contains the receiving of a request. Optionally, this activity is preceded by asking for advice. Although it is not mandatory to ask for advice, carrying out the 'ask for advice' activity enables a possible skip of the 'decide date publication' activity later on in the process.

Depending on the fact whether exemption is applicable (this is a decision based on local data), cooperation for this has to be given and consequently, the decision has to be integrated. Thus, either both activities 'cooperate' and 'integrate decision' are executed sequentially or both are skipped.

Then, two sub processes are executed in parallel; a document has to be registered and a destination plan has to be constructed. After carrying out these corresponding tasks, the case can be finished in multiple ways. The finish case sub process takes care of this.

The register document sub process can be completed by performing both checks and work for a letter in parallel before sending it. The checks themselves, a check for content and a check for safety, can also be performed concurrently. The letter on the other hand, has to be either a short or an extensive letter. Depending on the chosen version, an objection handling process, which for all cases is carried out at the end of the building permit process, takes a different amount of time.

To process the required destination plan, it only has to be modified if it already existed (which is the case in roughly 20% of the process executions), or a new plan has to be constructed. When a new destination plan has to be created, this can be done in one big, two medium or three smaller steps. If it will be performed in two steps this can be done in any order and at the same time.

Finally, to finish the building permit process, there are three possibilities. The process always sends the publication document. This can be enough to close the request, however, this activity can also be preceded by deciding a publication date or by deciding both a date and requiring a monument license. Recall that when the 'ask for advice' activity was skipped, the 'decide date publication' activity has to be carried out always. Moreover, the decision to execute activity 'receive monument license' is made locally and is not dependent on any previous decisions. After activity 'send publication document', a handle objection process has to be carried out. The duration of this is dependent of the letter that was chosen during the register document sub process.

The next paragraph highlights the various control-flow constructs/patterns present in this process model. Subsequently, Paragraph 6.2.3 elaborates on the resource and time distributions in the configurable process model.

6.2.2 Control-flow Patterns in the Model

To create models that represent similar organizations, variations of the process model explained in the previous paragraph have to be made. As Chapter 4 of this work discussed, this corresponds to having a different configuration setting for control-flow patterns in the model. In this section, the patterns present in the building permit process are described. For each pattern, several

locations of it are highlighted. The next section of this chapter then lists the concrete process model variants with certain varying patterns that are used for the remainder of this research.

- **Optionality**: choose to execute an activity or not

This pattern appears at multiple positions in the building permit process model. Especially the configuration option on occurs for a lot of activities, because this implies that all process instances contain the corresponding activity. Activities for which the configuration is set to optional are included in the 'exemption' sub process and shown in Figure 6.4. Note that in the general process model, no activities are turned off.

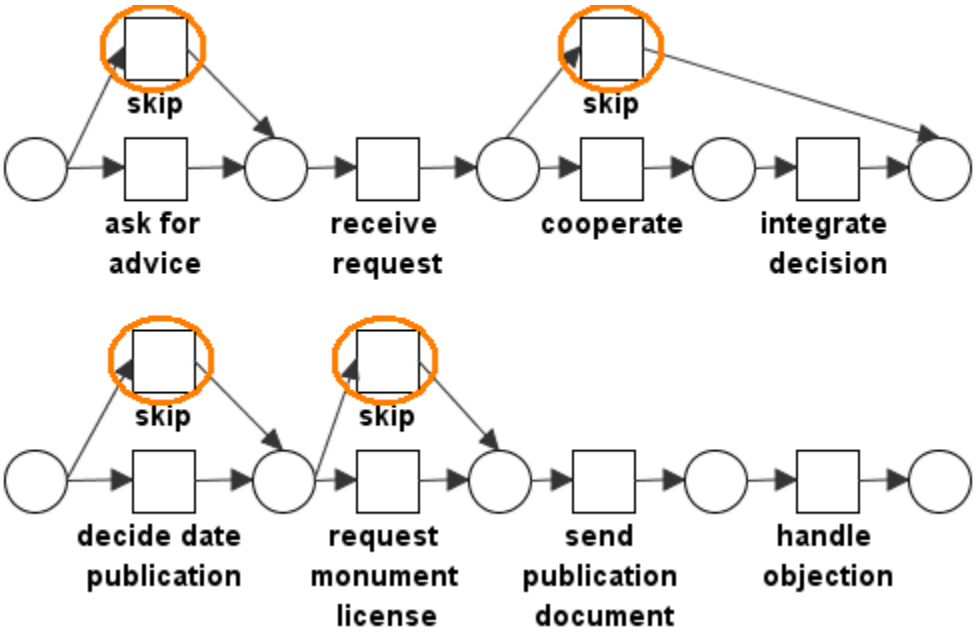

Figure 6.4: Optionality patterns configured as optional which appear in the model

- **Blocking**: choose to block (not execute) an activity

Blocking an activity in a branch leads to the fact that it cannot be continued; another branch has to be taken. Unfortunately, this pattern could not be introduced in the configurable process model, because the Coloured Petri net implementation used does not directly allow such a construct. For this reason, this pattern will be manually applied to determine the possible suitable locations in the model, its consequences for the model and its possible behaviour in the event log. Figure 6.5 shows the effect of the blocking pattern in this building permit process.

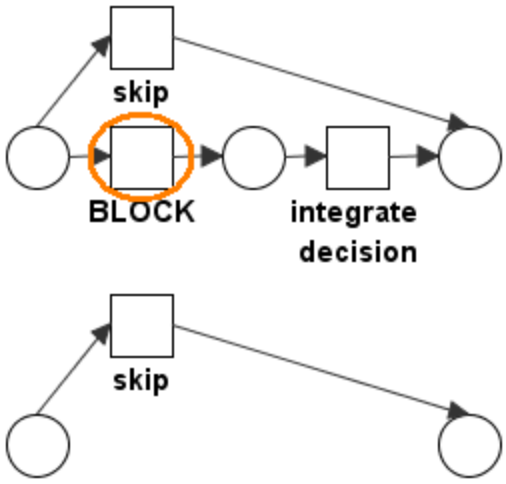

Figure 6.5: Blocking of an activity leading to more restricted behaviour

- **Hiding**: choose to hide an activity

In contrast to blocking, when hiding an activity, a certain path can still be continued, but the hidden activity will not be present in the event log. This pattern can be applied anywhere in the given configurable process model. However, at this moment, all configurations are turned off.

- **Interrelationship**: execution of an activity depends on the execution of another activity

In both the 'register document' and the 'make destination plan' sub processes, the interrelationship pattern occurs. If a short letter is written, no extensive letter is created and vice versa (mutually exclusive). Furthermore, if a new destination plan is made, an existing one is not modified and the other way around. These decisions are made separately for each process instance and are thus configured as optional. Remark that both these interrelationship patterns can be interpreted as an exclusive choice.

Additionally, activities 'check content' and 'check safety' occur together or they both do not occur (mutually dependent). The configuration option in the general model is set to on for this pattern. Also, activities 'make 2a plan' and make '2b plan' and the activities 'make 1/3 plan', 'make 2/3 plan' and 'make 3/3 plan' occur together or do not occur. Finally, when exemption is applicable, if 'cooperate' occurs then 'integrate decision' also occurs. In contrast to the checks, these decisions are set optional, such that each process instance makes the decision itself.

Some interrelationship patterns are highlighted in Figure 6.6. Mutually exclusive is depicted by blue colouring and mutually dependent by red colouring.

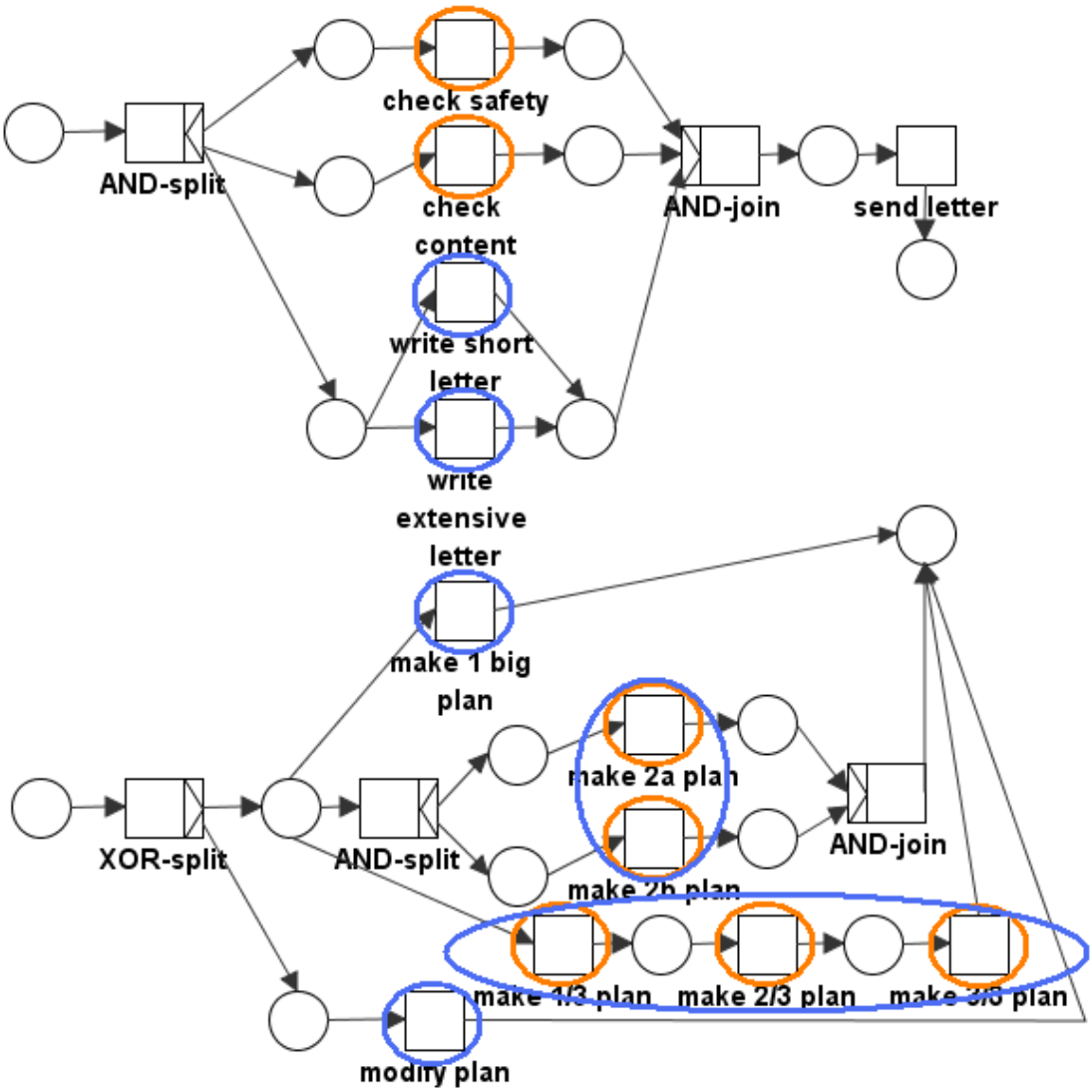

Figure 6.6: Interrelationship patterns present in model

- **Interleaved parallel routing**: choose in which order a set of activities is executed

This pattern allows for the execution of its associated activities in any order. Both the entire sub processes 'register document' and 'make destination plan' themselves happen in parallel. Inside these sub processes, the checks and the letter can be carried out interleaved. Furthermore, the checks themselves can be carried out in any order. All these decisions are made locally: for each case in the business process the order of the corresponding activities is decided separately.

- **Parallel split**: choose a subset of the activities

A building permit case is closed by either carrying out all activities of the sub process 'finish case', or by doing only a subset; this with the limitation that within the building permit process, the order of this subset is strict. See Figure 6.7

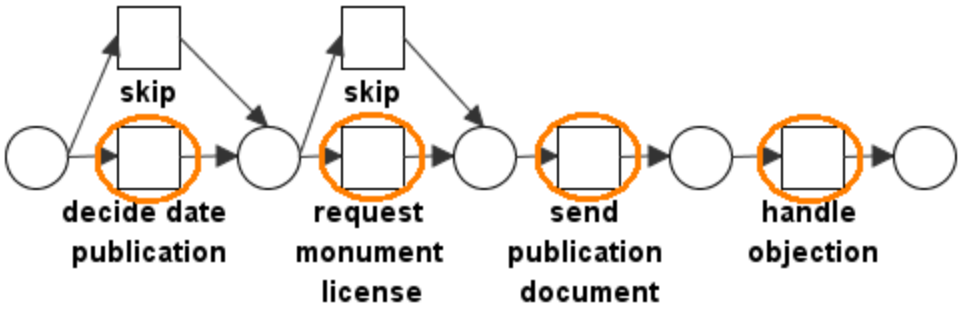

Figure 6.7: Parallel split patterns with a strict order

- **Exclusive choice**: choose exclusively an activity

The exclusive choice pattern is already introduced by the mutual exclusive instantiations of the interrelationship construct given before.

- **Aggregation**: combine activities into one activity

Clearly, the 'make destination plan' sub process embeds an aggregation pattern, see Figure 6.8. If a new plan has to be written, this is done by carrying out only the activity 'make 1 big plan', instead of 'make 2a plan' and 'make 2b plan' or activities 'make 1/3 plan', 'make 2/3 plan' and 'make 3/3 plan'. For each process instance, another aggregation level can be chosen. Therefore, this pattern is set to optional. In the general model, all other activities have an aggregation pattern that is set to off.

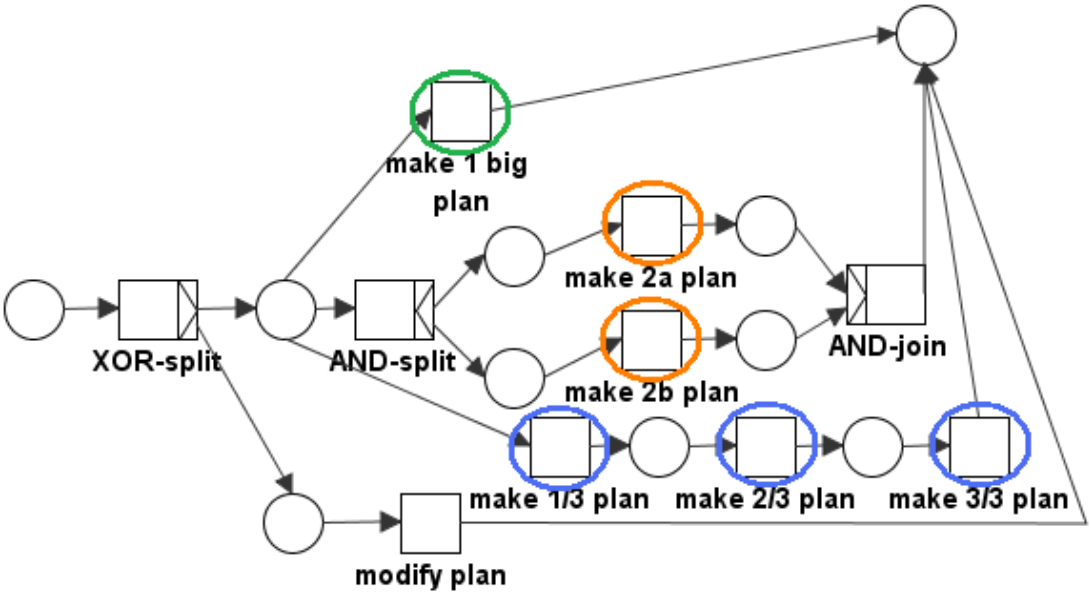

Figure 6.8: Aggregation patterns configured as optional which appear in the model

- Other patterns:
    - **Loop**: possibly carrying out one or more activities multiple times

    The loop pattern is not included in the configurable process model of handling building permits, because it is not suitable for the recommendation service. Since the recommendation algorithm looks at timing information to determine the best next action regarding reducing process times, it never recommends to redo certain activities, since this always takes more time than other possible behaviour.

6.2.3 Resource and Time Related Distributions in the Model

For the building permit business process, a number of decisions are made for the distributions of arrivals of new cases, the number and competences of resources, the duration of activities etc. As stated before, these resource and time related distributions remain unchanged for all process model variants that are derived from the general model. The reason for this is that variations of the business process model are created regarding control-flow patterns. Lower or higher levels of performance can then be explained by these changes in configuration settings. Furthermore, the research questions can be solved by these findings. However, by also changing the distributions of time and/or resources, additional alteration in performance can be caused, which makes it unclear whether performance improvement or deterioration was due to the changed control-flow or other resource and time distributions.

For the arrival of new cases in the process, every hour a Bernoulli distribution with parameter 0.5 as probability for success is executed. This means that there is 50% chance of success. If the experiment succeeds, two cases arrive in the building permit process; otherwise only one case will start the process.

A total of 25 employees are full-time available for carrying out the activities involved. Because of this number and the arrival distribution, (long) waiting times occur in the process model, as in real-life also occur. All employees can execute all activities. Moreover, they have the same duration distribution for these activities. This distribution is a normal distribution with the mean listed in Table 6.1 and a standard deviation of 0.5 hours. Such a distribution is chosen to incorporate variability in the model such that in some situations, a resource takes longer time to finish an activity than in other situations.

Note that the handle objection activity represents a process that takes place after sending the publication document. No resource is required for this. The duration of handling the objection is dependent of the letter: if a short letter was written, it takes between 2 and 6 hours, otherwise between 0 and 3 hours using a discrete distribution.

The next section describes the variations of the configurable process model discussed in terms of control-flow.

Activities	Duration in hours
cooperate, integrate decision, check safety, send letter, make 2a plan, make 2b plan, decide date publication, send publication document	2
make 1/3 plan, make 2/3 plan, make 3/3 plan	3
ask for advice, write short letter, request monument license	4
receive request, check content, modify plan	5
make 1 big plan	6
write extensive letter	7

Table 6.1: Duration times of activities

6.3 Process Model Variations

Because only 4 organizations are derived from the configurable process model, it is decided to keep the different types of control-flow patterns as separate as possible. This allows for deriving conclusions from changes in performance regarding these other configurations. In light of this, instead of using an iterative approach in which each organization contains an additional number of changes to the original configurable process model, a flower approach is used, as is visualized in Figure 6.9.

Organization 1 is the exact same process model as described in the previous section. The remaining 4 organizations are given below. For each variant, the changes from the 'general' configurable process model are indicated.

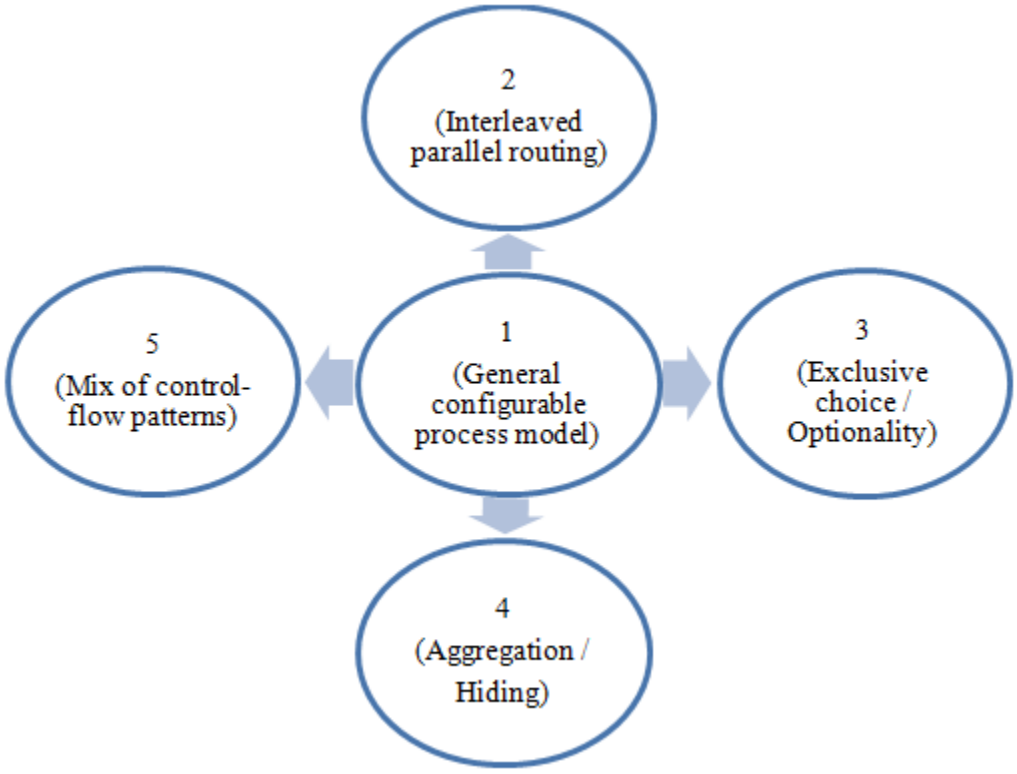

Figure 6.9: Organizations derived from configurable process model for experiments

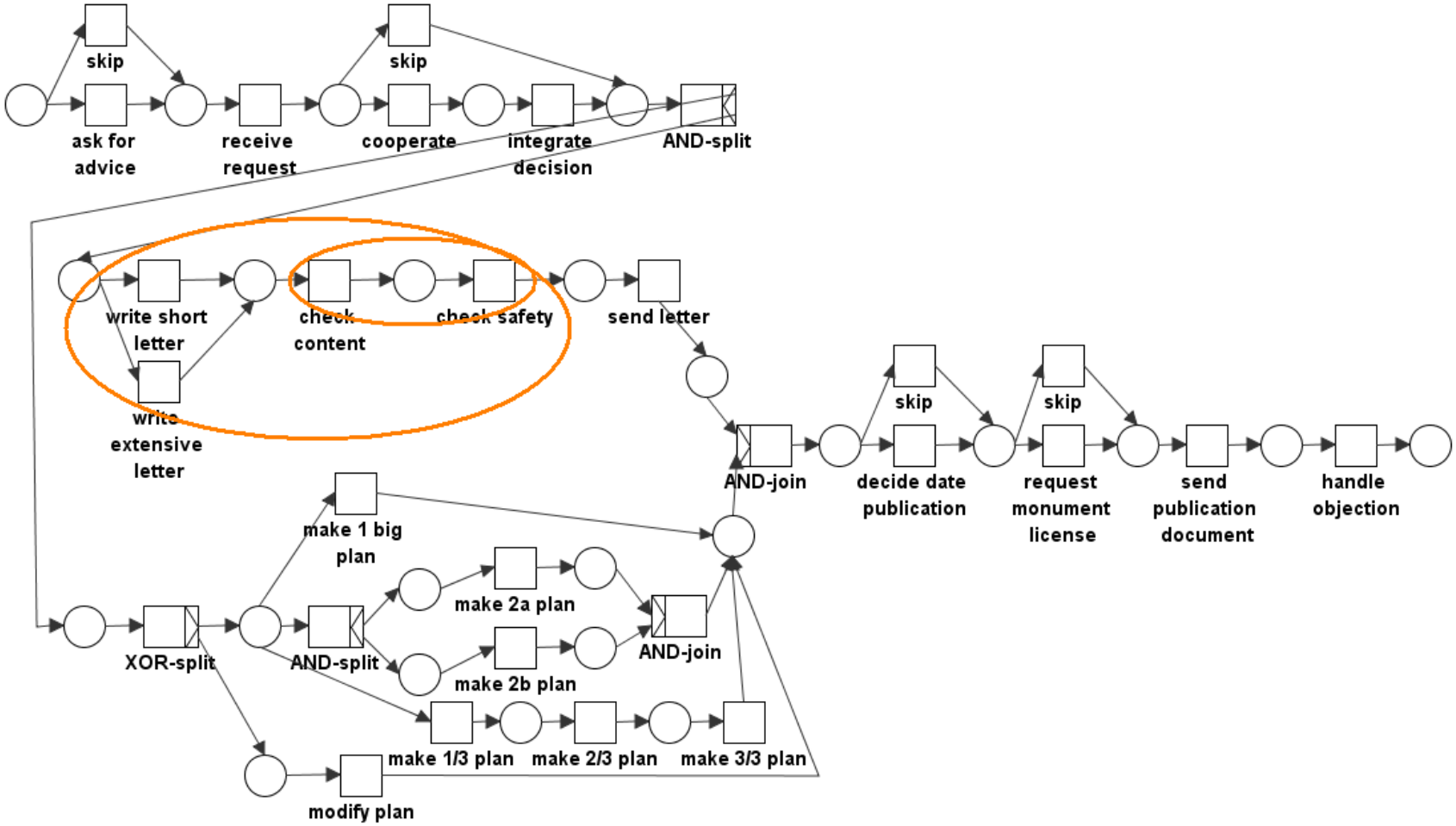

Figure 6.10: Organization 2

Compared to organization 1, changes with respect to the interleaved parallel routing configuration pattern are made. The order possibilities are reduced by the fact that the ‘register document’ sub process contains only sequential behaviour now. First either activity ‘write short letter’ or ‘write extensive letter’ has to be carried out before ‘check content’ and subsequent ‘check safety’ can occur.

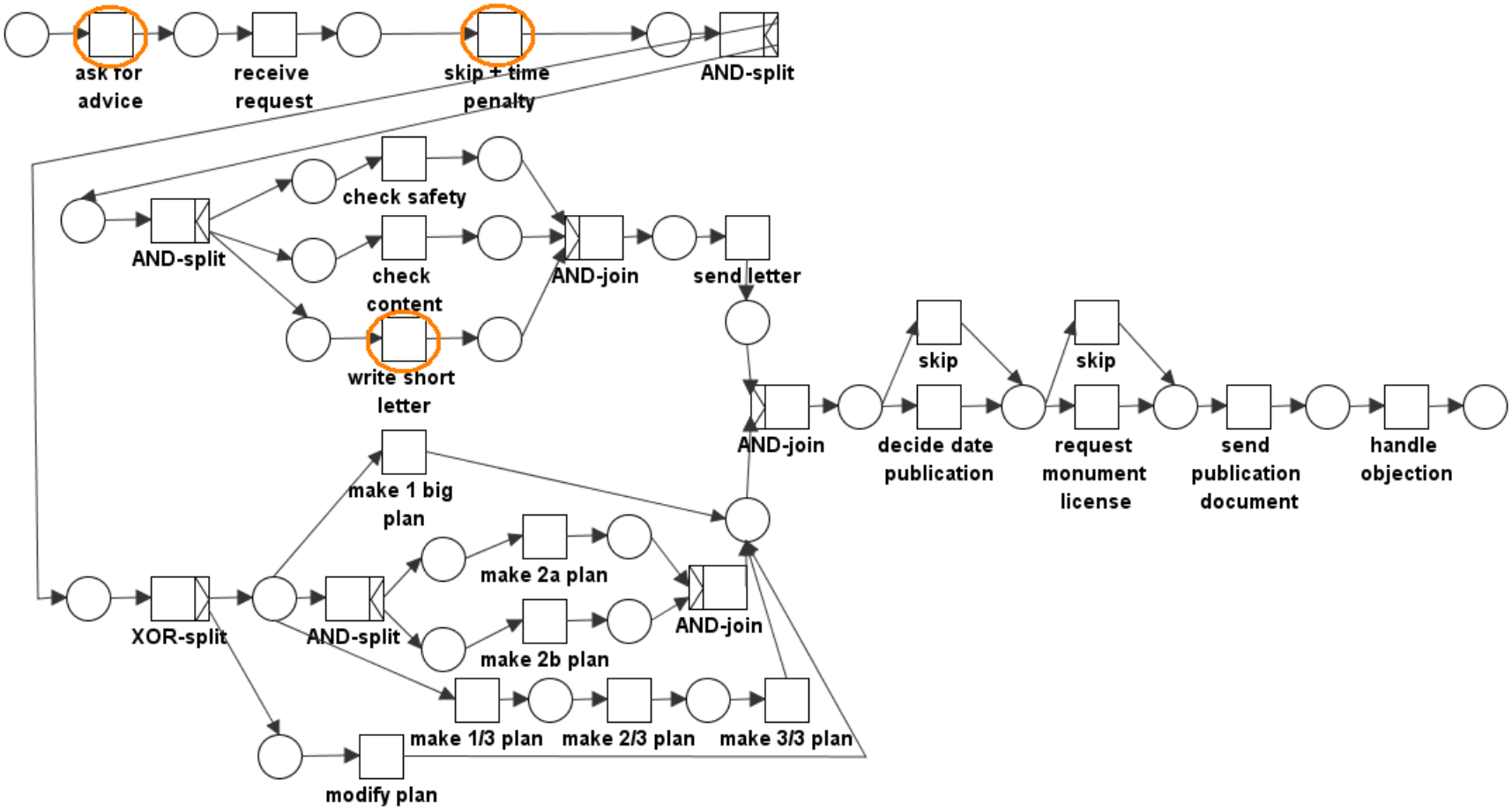

Figure 6.11: Organization 3

Organization 3 contains changes for the exclusive choice and optionality configuration patterns. The optionality configuration of activity ‘ask for advice’ is changed from optional to on: it has to be carried out for each case. Moreover, the ‘exemption’ sub process is always skipped, or in other words, switched off. This results in a time penalty for the ‘decide date publication’ activity; its duration will increase. Finally, there is no choice for a letter anymore; a short letter is the only option in this process model (‘write extensive letter’ is switched off).

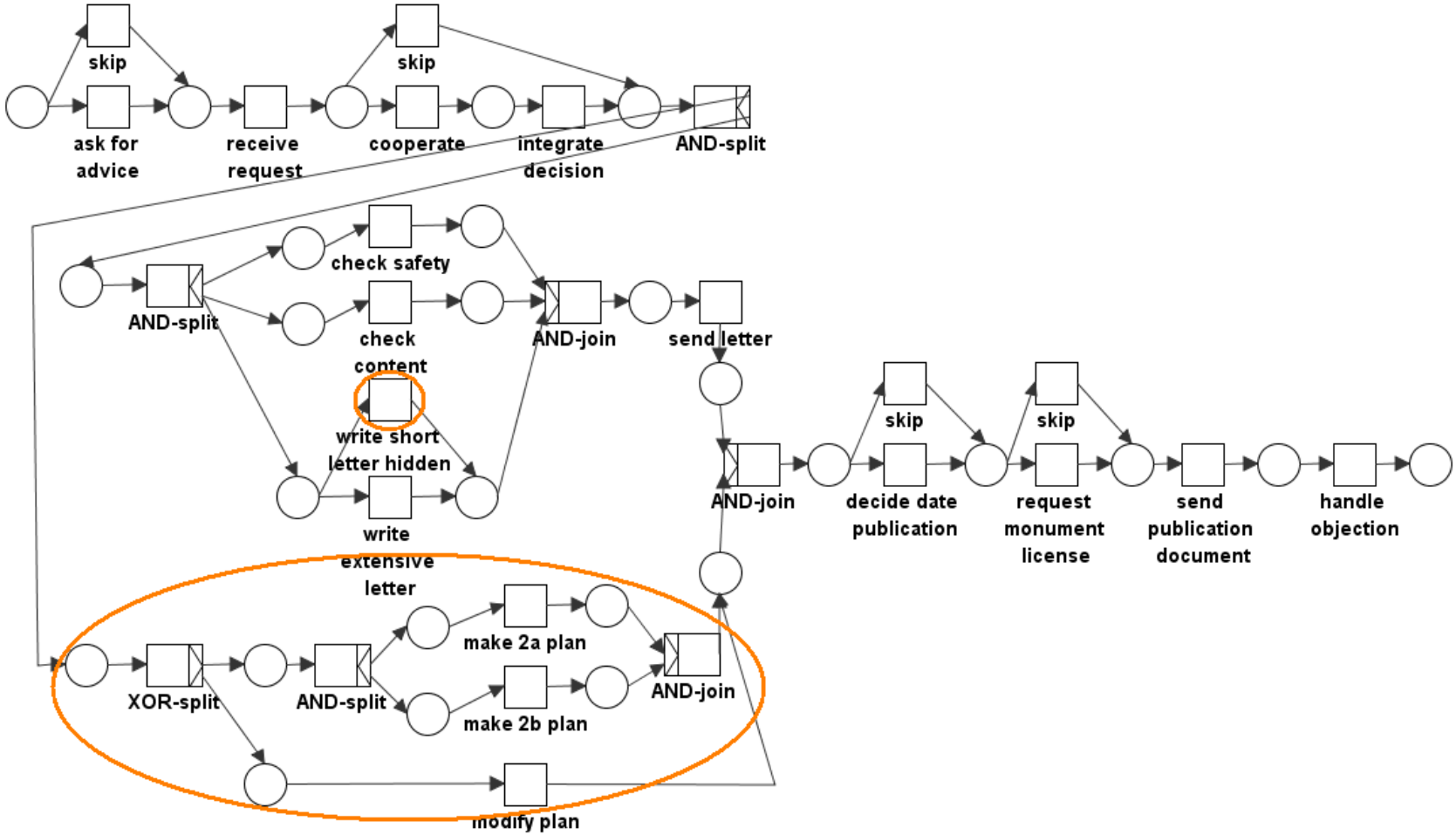

Figure 6.12: Organization 4

Aggregation and hiding are the control-flow patterns for which changes are made for organization 4 compared with organization 1. 'Write short letter' is hidden and will not be contained in the event log and recommendation requests/responses. Furthermore, a new destination plan has to be made in two steps: its aggregation level is 2 for all cases. Note that the chance of carrying out 'modify plan' for a case goes from 20% to 33%.

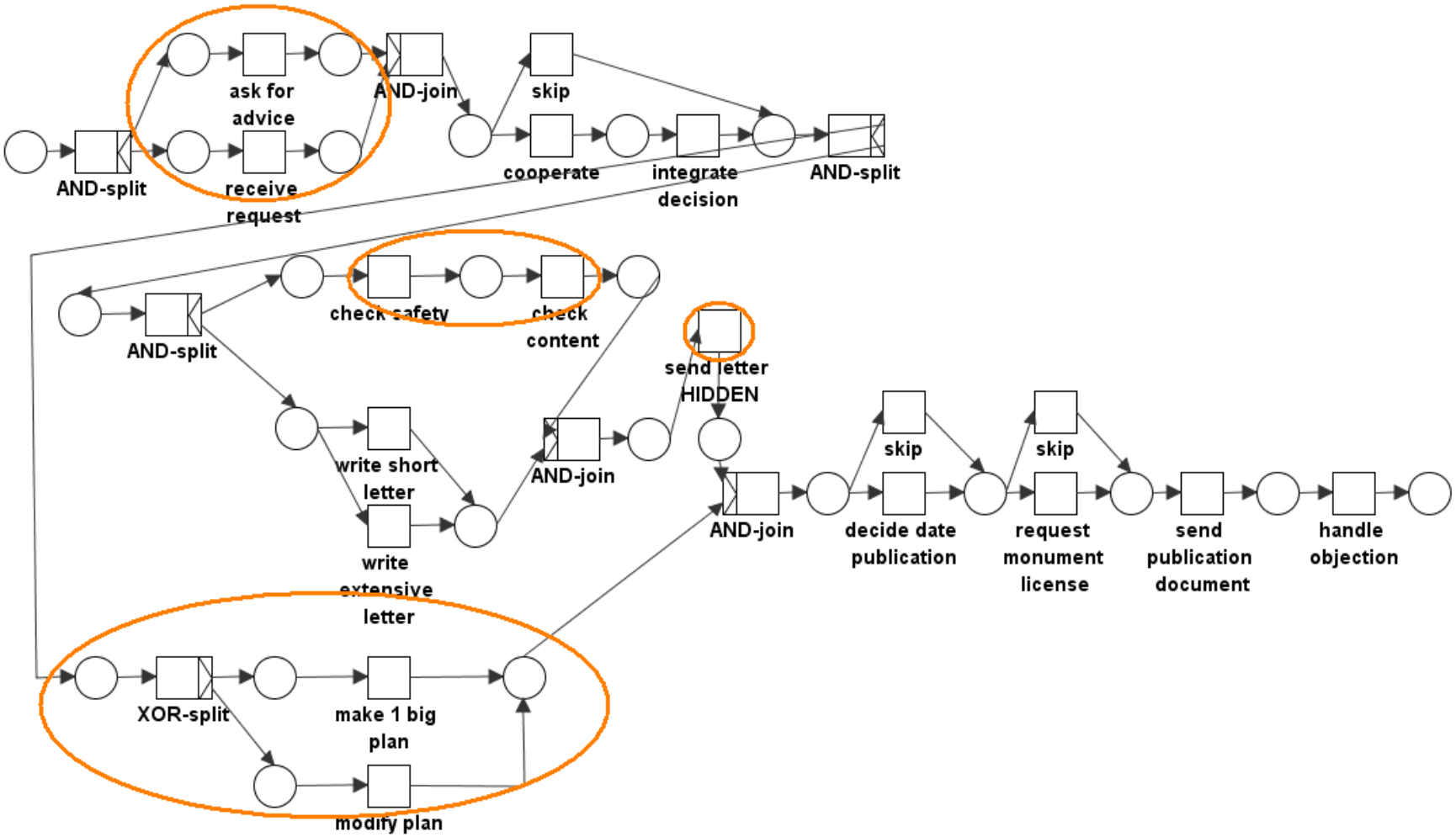

Figure 6.13: Organization 5

Organization 5 has a mix of changes compared to the process model of organization 1. Different control-flow patterns are configured in another way to test the effect of a larger number of changes. First of all, the aggregation level of the 'make destination plan' sub process is set to 1. It

always consists of only one step. Moreover, the ‘check safety’ activity has to precede the ‘check content’ activity, which decreases the possible order possibilities of cases. This is undone by the addition of parallel behaviour at the start of the case: ‘ask for advice’ and ‘receive request’ occur in parallel and thus become mutual dependent. The optionality of ‘ask for advice’ is changed from optional to on. Finally, ‘send letter’ is hidden. This control-flow pattern is located at another position than the hiding pattern of organization 4.

By making use of the implementation that was discussed in the previous chapter, experiments set up in Section 6.1, using historical data and executable process models of these 5 organizations, are executed. A number of performance related statistics and results are computed. These statistics and results are presented and interpreted in the next sections.

6.4 Experiment Statistics

To verify how many times the Multiple Log Recommender provided a recommendation with more certainty than random per experiment, the statistics listed in Table 6.2 are computed. These statistics are obtained using the following formula:

*100 – ((nr. of random recommendations / total nr. of recommendations) * 100)*

The averages over all experiments with the same percentage of recommendations for both transition system settings are computed. An observation that is made from these statistics is that the experiments with the transition system setting ‘Merge states with identical inflow’ yield better fitting results with respect to the specified percentage of non-random recommendations than the experiments with the transition system setting ‘No Merges’. For all experiments, the ratio seems correct, except for the experiment with process model 5 and event logs 1,2,3,4. However, the deviating numbers here are explained by the fact that the process model of organization 5 contains a relatively large degree of variation compared to the other models. Apparently, none the collective transition systems contains a fitting state accompanied by a transition equal to a possible activity in the model of organization 5, which resulted in a large number of random recommendations.

The fact that this transition system setting outperforms the other setting (‘No Merges’) has an advantage. It implies that the setting ’Merge states with identical inflow’ is the most suitable for this research, because more recommendation requests are responded by a non-random recommendation. Therefore, the obtained results will be more useful regarding answering the research questions, since they reflect the usage of own or cross-organizational historical data in as much situations as possible. This enables reducing experiments for the purpose of further research by 50%, because there is only one setting on the transition system domain remaining instead of two.

Additional statistics from the executed experiments, such as the exact of sequence, bag, set abstractions and random recommendations, can be found in Appendix A.

Organization	Event logs	No merges					Merge states with identical inflow				
		0%	25%	50%	75%	100%	0%	25%	50%	75%	100%
1	1	0	12.8	28.5	50.9	60.2	0	24.0	48.7	75.1	100
1	2345	0	17.8	27.7	47.9	54.6	0	24.9	49.0	73.3	100
2	2	0	21.3	35.9	64.7	93.4	0	23.3	50.9	75.1	100
2	1345	0	16.6	36.8	53.9	70	0	24.3	45.2	73.9	100
3	3	0	19.3	38.3	55.7	77.3	0	25.8	47.8	75	100
3	1245	0	15.3	35.3	55.8	84.7	0	23.3	50.4	77.1	100
4	4	0	18.7	38.4	61.4	71.4	0	24.8	50.9	78.7	100
4	1235	0	16.0	25	42.6	51.9	0	26.8	45.4	74.3	100
5	5	0	20.1	38.7	61.3	89.9	0	24.5	48.4	74.5	100
5	1234	0	8.4	18.9	24.7	32.9	0	11.3	18.0	22.1	31.1
Average		0	16.6	32.4	51.9	68.6	0	23.3	45.4	69.9	93.1

Table 6.2: Percentage of non-random recommendations

6.5 Experiment Results

This section presents the results of the experiments, which were set up in Section 6.1, with the organizational models of Section 6.3. The most important statistics of these experiments are given in the previous section and additional statistics are listed in Appendix A.

Recall that the goals of the experiments are twofold. Not only is researched whether it is possible to improve the performance of an organization by using historical data of other, similar organizations, but also when this would be the case, which particular control-flow configuration patterns play a positive or negative role in this process.

After studying the results of the experiments set up in Section 6.1, a high level of variability was discovered. These are caused by random behaviour, waiting times, and the fact that apparently, not enough cases are simulated in the experiments. To obtain more reliable results, it is chosen to extend the initial set of experiments. Each experiment is therefore carried out 10 times. This is chosen instead of executing an experiment with a simulation length which is 10 times larger, because for the latter option, most likely the waiting times increase for cases later on in the simulation. This is caused by the fact that a lower number of resources are available in the model than the required number of activities that can be carried out at the same time. As a consequence, this also affects the overall average flow times of cases. By keeping the same simulation length, this undesired side-effect is avoided.

The next paragraphs discuss the results of the experiments. Note that the complete set of results is listed in Appendix B.

6.5.1 Validation of Approach

As a confirmation that the approach using three transition systems and the recommendation algorithms of the Multiple Log Recommender works, the average flow times using percentages 0 and 100 are compared for experiments where organizations use their own historical data. Tables 6.3 and 6.4, and especially the graphs depicted in Figures 6.14 – 6.17, show a declining trend of average process times for all organizations using their own data, except organization 5.

The reason why for organization 5 no performance improvement is gained lies in the configuration of its control-flow patterns. These are configured such that it does not allow for making profit of the recommendation techniques. Compared to the default process model in organization 1, the aggregation pattern concerning the creation of a new destination plan is adapted. Furthermore, the interleaved parallel routing pattern concerning the checks in the 'register document' sub process is changed. Both these changes reduced the number of activities from which the Multiple Log Recommender can choose. As a result, it is prevented of making maximal use of the recommendation technique.

Organization	Merge states with identical inflow			
	Mean of sample	Standard Deviation of sample	Coefficient of Variation	95% Confidence Interval
1	6.072	0.357	0.059	0.221
2	6.400	0.372	0.058	0.230
3	6.204	0.362	0.058	0.224
4	6.590	0.418	0.063	0.259
5	5.533	0.257	0.046	0.159

Table 6.3: Probability measures of avg. flow times in days with 0% recommendations

Organization	Event logs	Merge states with identical inflow			
		Mean of sample	Standard Deviation of sample	Coefficient of Variation	95% Confidence Interval
1	1	5.036	0.392	0.078	0.243
2	2	4.728	0.188	0.040	0.116
3	3	4.123	0.100	0.024	0.062
4	4	5.405	0.354	0.065	0.219
5	5	5.863	0.308	0.053	0.191

Table 6.4: Probability measures of avg. flow times in days with 100% recommendations using own data

Organization	Event logs	Merge states with identical inflow			
		Mean of sample	Standard Deviation of sample	Coefficient of Variation	95% Confidence Interval
1	2345	4.197	0.302	0.072	0.187
2	1345	4.941	0.250	0.051	0.154
3	1245	3.989	0.194	0.049	0.120
4	1235	5.067	0.267	0.053	0.166
5	1234	5.588	0.366	0.066	0.227

Table 6.5: Probability measures of avg. flow times in days with 100% recommendations using cross-organizational data

6.5.2 Performance Improvement by Cross-Organizational Data

Next, the average flow times of completely random behaviour, listed in Table 6.3, are compared with the results of experiments in which 100% of the provided recommendations are followed while making use of cross-organizational data, which are presented in Table 6.5.

Again, as was the case when using own historical data, significant performance gains are reached for organizations 1 up to and including organization 4. These results answer the research question, because it shows that it is possible to improve performance using cross-organizational data of similar organizations.

Furthermore, it does not seem to matter that much if own data or data of all other organizations is used for performance improvements; see Figures 15 – 17. For organizations 2, 3 and 4, the confidence intervals overlap. This means that no statements can be made whether the use of own data is better than the use of historical data of the other organizations, or vice versa.

However, the results for organization 1 suggest that performance improvement can be gained by making use of cross-organizational data instead of own data. This is visualized in Figure 14. Using own data, the 95% confidence interval for the average flow times is 5.036 +/- 0.243 days. This means that with 95% probability, the average flow time of a new experiment with the same settings lies within the interval 4.793 – 5.279 days. Note that while using historical data of other organizations, a confidence interval of 95% for the average process times yields 4.197 +/- 0.187 days. Or in other words, a new business process execution of organization 1 following cross-organizational recommendations achieves with a probability of 95% average flow times within the interval of 4.010 – 4.384 days.

But, in order to claim that average process times for organization 1 are reduced using cross-organizational recommendations instead of recommendations based on own historical data, more experiments are needed to validate this. Such a strong statement requires additional statistics, which can be derived by carrying out a larger number of experiments where each experiment makes use of completely independently distributed data. This means that the process of generating event logs of organizations, described in Section 5.2, has to be repeated for every single experiment. Unfortunately, this is impossible for this research due to the amount of time this requires.

Yet, the results suggest that organization 1, in contrast to the other four organizations, can achieve better performance by making use of historical data from others instead of own data. A possible explanation for this is that it contains enough possibilities for the use of recommendations. More than the other organizations, it allows for parallel behaviour with a relatively higher number of activities to choose from, because of occurrence of the control-flow configurations optionality, interleaved parallel routing, exclusive choice and aggregation.

6.5.3 General Evolution of Performance

The previous paragraphs described that for all organizations, except organization 5, performance improvement is reached by following all recommendations of the Multiple Log Recommender. Furthermore, in general it does not matter whether own historical data or cross-organizational data is used. This paragraph elaborates on the changes in average flow times while navigating from 0 to 100% recommendations. These trends for organizations 1 – 5 are visualized in Figures 14 – 17 respectively.

It could be expected that performance improves gradually when making use of a higher percentage of recommendations. However, this only seems to be the case in the experiments for organization 3. Figure 6.16 shows the declining trend for all percentages of organization 3. The corresponding probability measures are listed in Table 6.8.

A possible reason is that organization 3 is the only organization in which the exemption sub process is always skipped. In the other organizations, the decision whether or not this sub process is executed for a process instance is made locally. The percentage of recommendations has no influence on this. Therefore, there is higher randomness and variability in the performance of process executions.

For all other organizations, there are confidence intervals of percentages 25, 50 and 75 that overlap with lower percentages. This means that it cannot be stated that the average process times in days improve when less random recommendations are provided.

Organization 2 shows a declining trend for the experiments using its own data, however for the experiments using cross-organizational data, actually a deterioration of performance is observed. For organizations 1 and 4, it is the other way around. A declining trend occurs for experiments that make use of other data, but not for the experiments using own historical data. Finally, the results of the experiments with organization 5 are visualized in Figure 6.18 and listed in Table 6.10. Recall that for this organization, improvement is not possible.

Another explanation of the fact that there is no gradual improvement is the sensibility of the Multiple Log Recommender. It seems when somewhere for a process instance a recommendation is neglected, and thus another activity is executed, this can have disastrous consequences regarding performance, possibly even worse than completely random. The reason for this is the fact that the recommendation algorithm cannot handle partial process instances which only slightly deviate from the best observed behaviour. By neglecting a recommendation, it is possible that another part of the collective transition system, which not performs that well, conforms to the process instance and is followed and used to base recommendations on. Remark that this drawback is inherent to the underlying technique of recommendations, described in Section 3.2, because future behaviour of partial process instances is steered by historical data.

% recomm.	Event logs	Merge states with identical inflow			
		Mean of sample	Standard Deviation of sample	Coefficient of Variation	95% Confidence Interval
0	-	6.072	0.357	0.059	0.221
25	1	6.261	0.352	0.056	0.218
25	2345	6.295	0.319	0.051	0.198
50	1	6.337	0.249	0.039	0.154
50	2345	6.030	0.490	0.081	0.304
75	1	6.361	0.450	0.071	0.279
75	2345	5.662	0.287	0.051	0.178
100	1	5.036	0.392	0.078	0.243
100	2345	4.197	0.302	0.072	0.187

Table 6.6: Probability measures of avg. flow times in days of experiments for organization 1

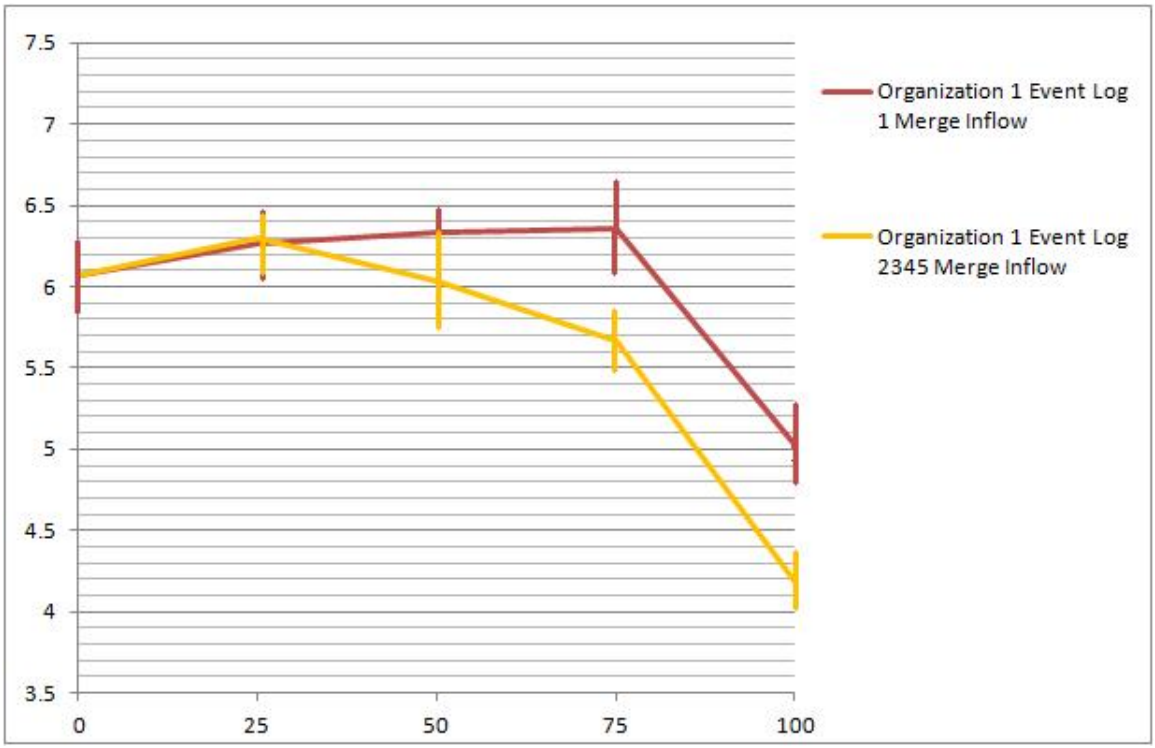

Figure 6.14: Avg. flow times in days with confidence intervals for experiments organization 1

% recomm.	Event logs	Merge states with identical inflow			
		Mean of sample	Standard Deviation of sample	Coefficient of Variation	95% Confidence Interval
0	-	6.400	0.372	0.058	0.230
25	2	6.184	0.356	0.058	0.221
25	1345	6.150	0.449	0.073	0.278
50	2	5.858	0.380	0.065	0.235
50	1345	6.477	0.414	0.064	0.257
75	2	5.656	0.461	0.082	0.286
75	1345	6.165	0.647	0.105	0.401
100	2	4.728	0.188	0.040	0.116
100	1345	4.941	0.250	0.051	0.154

Table 6.7: Probability measures of avg. flow times in days of experiments for organization 2

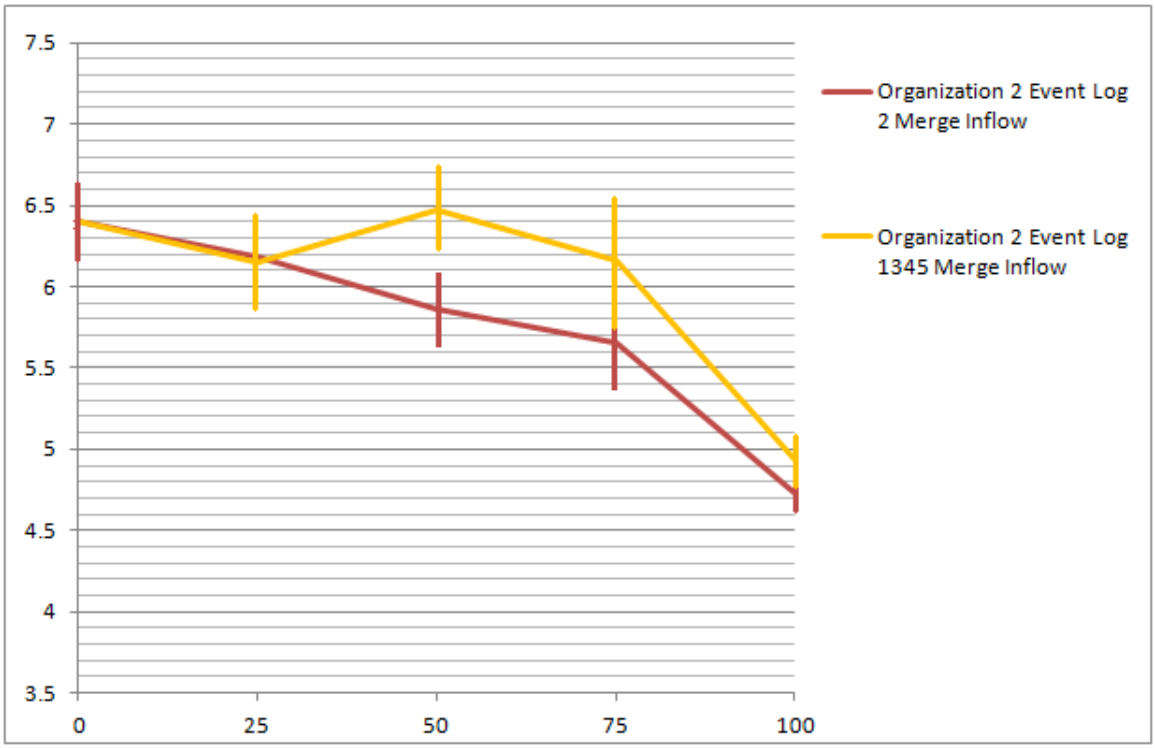

Figure 6.15: Avg. flow times in days with confidence intervals for experiments organization 2

% recomm.	Event logs	Merge states with identical inflow			
		Mean of sample	Standard Deviation of sample	Coefficient of Variation	95% Confidence Interval
0	-	6.204	0.362	0.058	0.224
25	3	5.601	0.211	0.038	0.131
25	1245	5.828	0.274	0.047	0.170
50	3	5.143	0.284	0.055	0.176
50	1245	5.270	0.214	0.041	0.133
75	3	4.734	0.222	0.047	0.138
75	1245	4.669	0.161	0.034	0.100
100	3	4.123	0.100	0.024	0.062
100	1245	3.989	0.194	0.049	0.120

Table 6.8: Probability measures of avg. flow times in days of experiments for organization 3

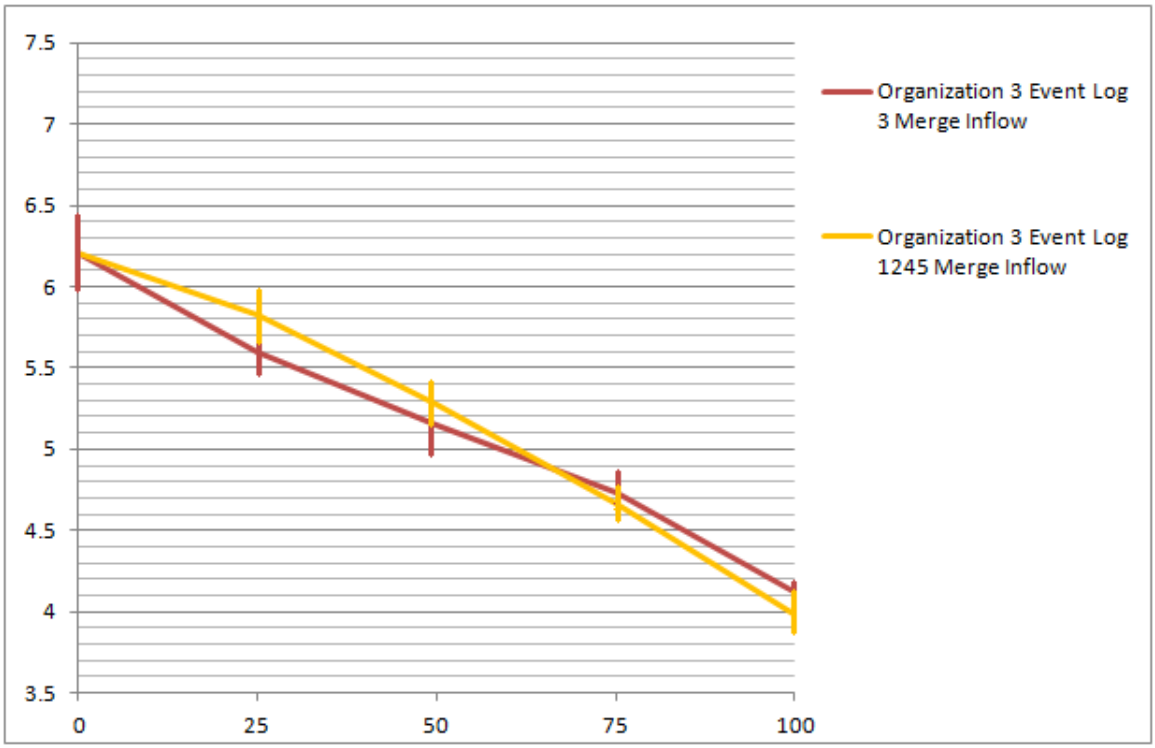

Figure 6.16: Avg. flow times in days with confidence intervals for experiments organization 3

% recomm.	Event logs	Merge states with identical inflow			
		Mean of sample	Standard Deviation of sample	Coefficient of Variation	95% Confidence Interval
0	-	6.590	0.418	0.063	0.259
25	4	6.409	0.364	0.057	0.226
25	1235	6.474	0.321	0.050	0.199
50	4	6.772	0.243	0.036	0.150
50	1235	6.429	0.268	0.042	0.166
75	4	6.755	0.737	0.109	0.457
75	1235	6.436	0.650	0.101	0.403
100	4	5.405	0.354	0.065	0.219
100	1235	5.067	0.267	0.053	0.166

Table 6.9: Probability measures of avg. flow times in days of experiments for organization 4

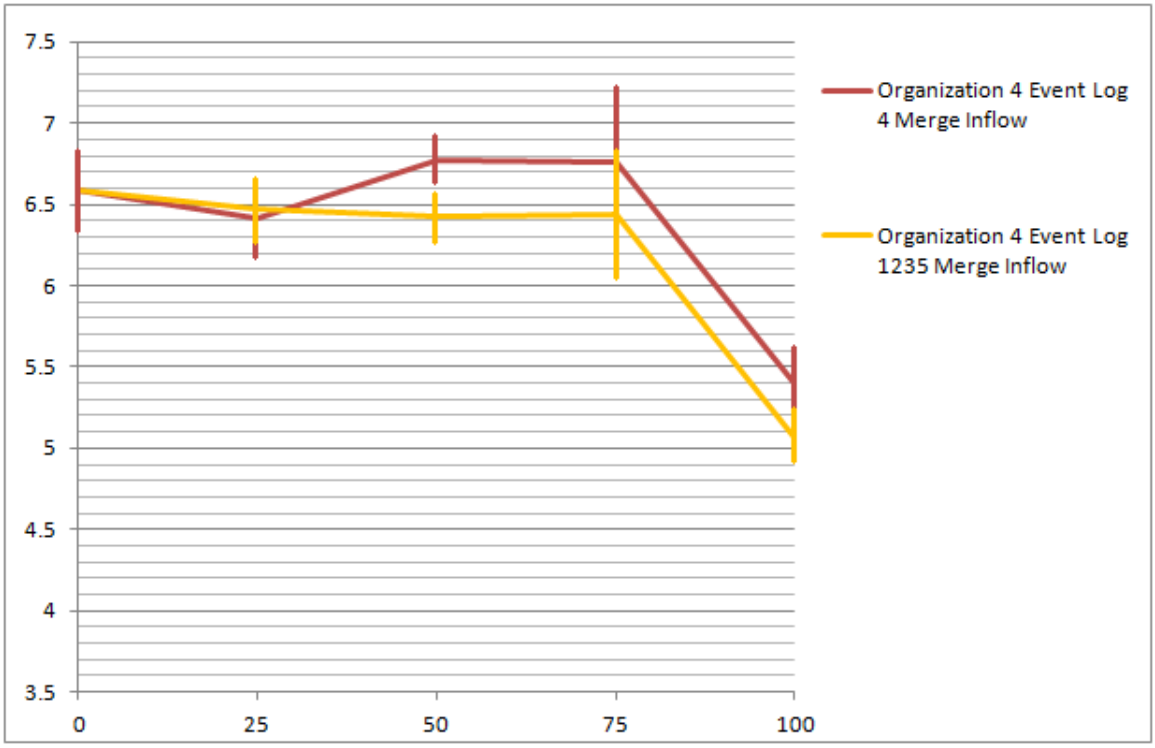

Figure 6.17: Avg. flow times in days with confidence intervals for experiments organization 4

% recomm.	Event logs	Merge states with identical inflow			
		Mean of sample	Standard Deviation of sample	Coefficient of Variation	95% Confidence Interval
0	-	5.533	0.257	0.046	0.159
25	5	5.619	0.308	0.055	0.191
25	1234	5.438	0.282	0.052	0.175
50	5	5.616	0.194	0.035	0.120
50	1234	5.504	0.323	0.059	0.200
75	5	5.821	0.356	0.061	0.220
75	1234	5.304	0.369	0.070	0.229
100	5	5.863	0.308	0.053	0.191
100	1234	5.588	0.366	0.066	0.227

Table 6.10: Probability measures of avg. flow times in days of experiments for organization 5

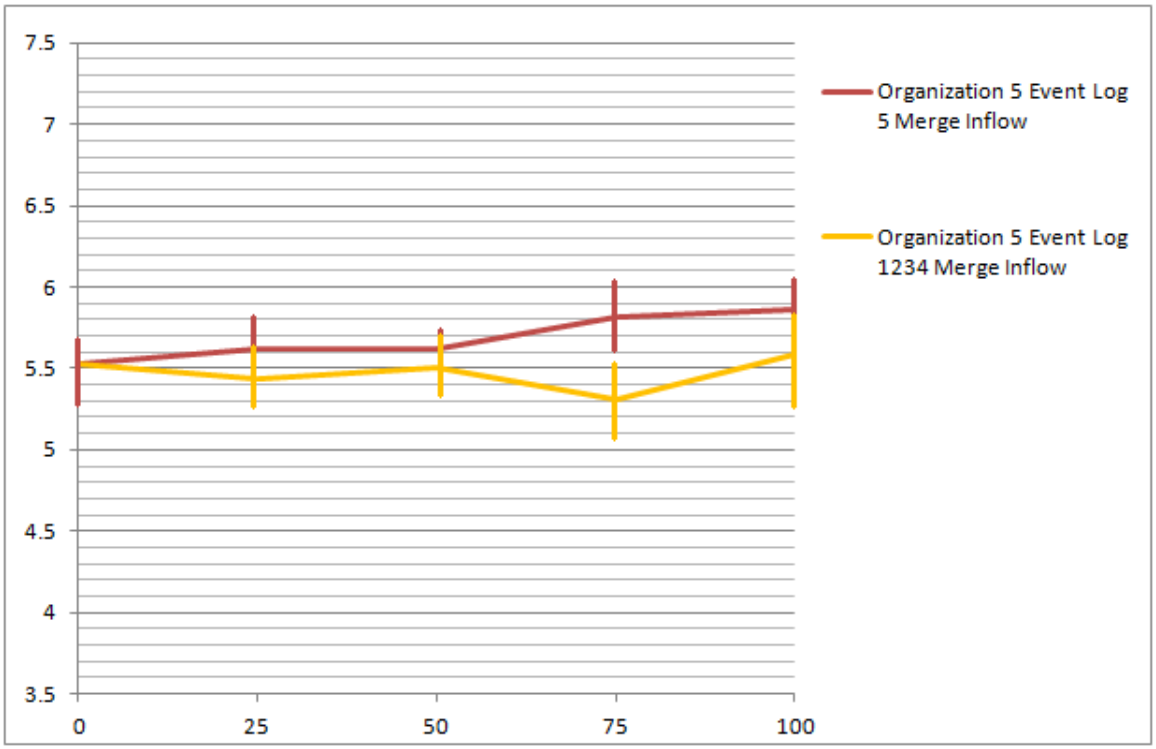

Figure 6.18: Avg. flow times in days with confidence intervals for experiments organization 5

This chapter presented statistics and results of experiments using process models containing different control-flow pattern configurations. For the transition system domain, the setting ‘Merge states with identical inflow’ appeared to be most suitable for this research. It allows for the highest percentage of recommendation requests which can be responded by non-random recommendations.

The results of the experiments show that for all organizations the average flow times in days significantly improves when all recommendations are followed compared to random behaviour, except for organization 5. From this finding can be concluded that certain control-flow pattern configurations have a negative impact on the usage of recommendations, and therefore performance improvement. More specifically, these are the configurations that limit the number of choices and activities that can be carried out in parallel. Among these are the removals of interleaved parallel routing, optionality, choice and aggregation patterns.

In general, for organizations which are configured such that they do possess these patterns sufficiently, it appears that significant performance improvement can be gained, independent of the fact whether own or cross-organizational data is used. These findings cover sub question 4.

The next chapter presents conclusions about this research, by discussing answers to the research questions. Moreover, possibilities for future work are indicated.

Chapter 7
Conclusions

This master thesis described the problem that there is no work that provides information whether performance improvement is possible by making use of cross-organizational data of similar business processes. From this problem description, a research question and sub questions were composed which are repeated below.

> ***Research question:*** *Can organizations improve their performance by making use of historical data of other, similar organizations?*
>
> ***Sub question 1:*** *How can historical data be used to improve performance?*
>
> ***Sub question 2:*** *How can historical data of different organizations be combined such that it can be used for performance improvements?*
>
> ***Sub question 3:*** *What are similar business processes and which differences in control-flow have an impact on the usability of the techniques for performance improvement?*
>
> ***Sub question 4:*** *What is the difference between using historical data of the same or other, similar organizations?*

In this concluding chapter, a reflection on the approach is provided as well as a discussion whether and to what extent the research questions are addressed. Furthermore, Section 7.2 lists the deliverables that were created during this master project. Finally, possibilities for future work are indicated in Section 7.3.

7.1 Discussion

After elaborating on related work in Chapter 2, Chapter 3 discussed an existing framework called Operational Support and specifies that by making use of recommendations, historical data can be used for performance improvements. At the same time, limitations of this approach for this research were indicated. By extending the process model which is used for providing

recommendations, a solution is introduced that combines historical data of multiple organizations in a collective transition system. In this way, cross-organizational data can be used in order to improve performance. With these steps, sub questions 1 and 2 are covered.

The problem defined by the third sub question is solved in Chapter 4. A notion of similar organizations is obtained by making use of configurable process models. These specific types of process models allow for configuring different control-flow patterns for process models which are derived from such a configurable process model. The resulting model has the same vocabulary as the configurable process model. Furthermore, the implications for making use of recommendations through different configurations of control-flow patterns are elaborated on.

Chapter 5 explained the implementation of this research. Plug-ins in ProM were created which implement the collective transition system and a service called Multiple Log Recommender that provides recommendations using this process model, by making use of the Operational Support Framework. Additionally, executable process models were constructed that record historical data in an event log and communicate recommendation requests and responses with the Multiple Log Recommender. The performance and statistics of business process executions in such a model furthermore are measured and calculated by a ProM plugin. This plug-in automatically starts experiments using varying configurations for the collective transition system with respect to historical data and transition system settings, the executable process model and a percentage of recommendations which should be followed.

In Chapter 6, a set of such experiments is set up. To cover sub question 4, a business process inspired by a real-life process is modelled as a configurable process model. From this model, 4 other process models are derived that represent similar organizations. For each organization, a number of control-flow patterns are configured in a different way, such that performance increases or decreases can be linked to a specific change in control-flow configurations. From the statistics and results of the execution of the experiments, a number of conclusions are drawn. First of all and most importantly, the results show that it indeed is possible to improve performance of an organization by making use of cross-organizational data of similar organizations. By this conclusion, the research question of this master thesis is answered and the corresponding problem description is solved. However, additional observations are made which elaborate on the impact of control-flow patterns and other settings which are involved in more detail.

As a first remark, it turned out that the Multiple Log Recommender is very sensitive to random behaviour. That is, the moment a user neglects a recommendation and chooses to carry out another activity instead, the results show that performance can decrease substantially and become as worse as process instances which are executed ‘totally random’. The underlying reason is a consequence of the fact that the Multiple Log Recommender is based on the recommendation technique of the Operational Support Framework. Typically, this technique steers on observed behaviour. A sudden deviation from a path by taking another transition might have as a consequence that the case is identified with a state that performed relatively bad and this state is used for further recommendations. Thus, users of the Multiple Log Recommender are strongly discouraged to ignore the recommendations, because this can have a serious negative impact on the performance.

Furthermore, these effects should also be seen in light of the process models that were used. Most probably, for larger models with a wider variety of choices and a higher number of

recommendation requests per process instance, this sensibility will further increase, because there is a higher chance of diverging from the best path in the collective transition system when carrying out a random recommendation.

Another interesting observation is the fact that there are organizations which contain control-flow configurations for which the Multiple Log Recommender cannot provide performance gains. In particular, when there simply are not enough choices at the same time possible, there is no room for improvement by the recommendation service. Typically, they are caused by a relatively low number of occurrences of optional interrelationship, local interleaved optionality, exclusive choice, and aggregation configuration settings.

In light of this, for organizations which do possess such control-flow configuration patterns, the results imply that comparable average flow times are obtained using combined historical data of other organizations instead of own data recordings. This finding addresses sub question 4. In both these scenarios the average process times are improved while making use of the Multiple Log Recommender instead of random decisions. As such, substantial performance gains are achieved using own or cross-organizational historical data.

There was one organization for which the use of cross-organizational data actually resulted in performance improvement compared to using its own data. This suggests that under some circumstances, the use of cross-organizational data yields better results than own historical data. However, more experiments are needed to validate this strong statement. It is one of the possibilities for further research that are recommended in Section 7.3.

The next section first lists the deliverables of the implementation phase of this work. Then, Section 7.3 elaborates on possibilities for future work that can dive in even more detail regarding the occurrence of control-flow configuration patterns separately in both the business processes of the process execution and the collective transition system.

7.2 Deliverables

The implementation of this research consists of a number of executable models and applications. These are listed below:

Collective transition system plug-in

A ProM plug-in which takes multiple event logs as input and creates an annotated, collective transition system as output.

Multiple Log Recommender plug-in

A ProM plug-in able to provide recommendations to process instances of a PAIS by making use of three collective transition systems.

Experiment plug-in

A ProM plug-in for simulating experiments using different settings and computation of performance related statistics and results.

Executable model for event log

A CPN model of an organization for the creation of an event log

Executable model for control-flow recommendations

A CPN model of an organization able to request and process control-flow recommendations

7.3 Future Work

This master project serves as a starting point for research related to performance improvement by making use of cross-organizational historical data. Due to the availability of resources and time, a number of possible research directions remain open. These possibilities for future work are stated below.

First of all, additional experiments can be carried out to further study the impact of control-flow patterns on performance by looking at the possible impact of position and occurrence of patterns. Also, to validate the finding that for an organization, better performance is achieved using cross-organizational data than by using own historical data, more experiments are required. If possible, these experiments should be performed with completely independently distributed data.

Secondly, besides the control-flow dimension, the effect of different resource patterns can be investigated. Examples of these are resource patterns with other duration distributions, varying competences, policies, work schedules, or varying number of resources etc. The software components of the implementation of this work are already adapted to enable such research. A CPN model able to request and process recommendations of both control-flow and resources is elaborated on in Appendix C. Furthermore, changes to the Multiple Log Recommender and Collective Transition System Miner are described.

Third, a larger organizational model can be used to incorporate the functionality of requesting and processing recommendations to investigate the impact of the number of recommendations and possible activities. This also allows for the implementation of filtering certain activities or looking only at a finite horizon. These techniques were not needed or applicable due to the size of the models that were studied in this research.

Another possibility for future work is the implementation of information retrieval related techniques which can be used for different naming conventions. This would have as a consequence that the similarity constraint of process models is made more flexible, since the same vocabulary does not need to be used. As such, process models of organizations can be used which contain deviating labels for an activity, which semantically represent the same activity.

Additionally, instead of looking at process times, other objective functions can be used, such as quality or costs. These objectives can relatively easily be implemented, because the implemented process times in the collective transition system, executable process models and Multiple Log Recommender can be replaced by the chosen objective's counterparts.

Bibliography

[1] *The New Industrial Engineering: Information Technology and Business Process Redesign.* **T.H. Davenport, and J.E. Short.** 4, Massachusetts : Massachusetts Institute of Technology, 1990, Vol. 31.

[2] *Patterns for Process-Aware Information Systems.* **Mulyar, N.A.** Eindhoven, 2009.

[3] *Process Mining: Discovery, Conformance and Enhancement of Business Processes:* Introduction.. **Aalst, W.M.P. van der.** Eindhoven : Springer-Verlag Berlin Heidelberg, 2011, pp. 1-25.

[4] *Process Mining: Discovery, Conformance and Enhancement of Business Processes:* Process Modeling and Analysis. **Aalst, W.M.P. van der**. Eindhoven : Springer-Verlag Berlin Heidelberg, 2011, pp. 29-57.

[5] *Coloured Petri Nets Modelling and Validation of Concurrent Systems:* Introduction to Modeling and Validation. **K. Jensen, and L.M. Kristensen.** Aarhus : Springer-Verlag Berlin Heidelberg, 2009, pp. 1-12.

[6] *Process Mining: Using CPN Tools to Create Test Logs for Mining Algorithms.* **A.K. Alves de Medeiros, and C.W. Günther.** DAIMI, Eindhoven, 2005, Proceedings of the Sixth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools, Vol. 576.

[7] *A Meta Model for Process Mining Data.* **B.F. van Dongen, and W.M.P. van der Aalst.** Eindhoven : s.n., 2005, In Conference on Advanced Information Systems Engineering, Vol. 161.

[8] *Beyond Process Mining: From the Past to Present and Future.* **W.M.P. van der Aalst, M. Pesic, and M. Song.** Eindhoven : Springer, 2010, Lecture Notes in Computer Science, Vol. 6051/2010, pp. 38-52.

[9] *Supporting Flexible Processes Through Recommendations Based on History.* **B. Weber, B.F. van Dongen, M. Pesic, C.W. Guenther, and W.M.P. van der Aalst.** Eindhoven, 2007.

[10] *Configurable Process Models as a Basis for Reference Modeling.* **W.M.P. van der Aalst, A. Dreilling, F. Gottschalk, M. Rosemann, and M.H. Jansen-Vullers.** Eindhoven : Springer-Verlag Berlin Heidelberg, 2006, BPM 2005 Workshops, pp. 512-518.

[11] *Configurable Process Models: Experiences from a Municipality Case Study.* **F. Gottschalk, T.A.C Wagemakers, M.H. Jansen-Vullers, W.M.P. van der Aalst, and M. La Rosa.** Eindhoven : Springer-Verlag Berlin Heidelberg, 2009, Lecture Notes in Computer Science, Vol. 5565/2009, pp. 486-500.

[12] *Model-based software configuration: patterns and languages.* **A. Dreiling, M. Rosemann, W. van der Aalst, L. Heuser, and K. Schulz.** 15, Münster, 2006, European Journal of Information Systems, pp. 583-600.

[13] *Configurable Process Models: A Foundational Approach.* **F. Gottschalk, W.M.P. van der Aalst, and M.H. Jansen-Vullers.** Eindhoven : Springer-Verlag Berlin Heidelberg, 2007, pp. 59-77.

[14] *Process Viewing Patterns.* **D. Schumm, F. Leymann, and A. Streule.** Stuttgart : IEEE Computer Science, 2010, pp. 89-98.

[15] *Process Mining Framework for Software Processes.* **V. Rubin, C.W. Günther, W.M.P. van der Aalst, E. Kindler, B.F. van Dongen and W. Schäfer.** Eindhoven : Springer-Verlag Berlin Heidelberg, 2007, Lecture Notes in Computer Science, Vol. 4470, pp. 169-181.

[16] *Access/CPN 2.0: A High-level Interface to Coloured Petri Net Models.* **Westergaard, M.** Eindhoven : Springer-Verlag Berlin Heidelberg, 2011, Lecture Notes in Computer Science, Vol. 6709, pp. 328-337.

[17] *CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets.* **A.V. Ratzer, L. Wells, H.M. Lassen, M. Laursen, J.F. Qvortrup, M.S. Stissing, M. Westergaard, S. Christensen, and K. Jensen.** Aarhus : Springer-Verlag Berlin Heidelberg, 2003, Lecture Notes in Computer Science, Vol. 2679, pp. 450-462.

[18] *A Generic Import Framework for Process Event Logs.* **C.W. Günther, and W.M.P. van der Aalst.** Eindhoven : Springer-Verlag Berlin Heidelberg, 2006, Lecture Notes in Computer Science, Vol. 4103/2006, pp. 81-92.

Appendix A

Additional Experiment Statistics

Organization	Event logs	Percentage	No merges				Merge states with identical inflow			
			Sequence	Bag	Set	Random	Sequence	Bag	Set	Random
1	1	0	0	0	0	795	0	0	0	826
		25	82	7	10	672	187	7	0	614
		50	178	33	18	575	378	4	3	405
		75	306	90	36	416	601	10	4	204
		100	331	82	34	295	740	4	0	0
1	2345	0	0	0	0	853	0	0	0	820
		25	114	21	10	668	186	3	0	569
		50	163	24	13	522	392	5	0	414
		75	300	39	20	390	537	4	0	197
		100	318	12	37	305	711	0	0	0
2	2	0	0	0	0	825	0	0	0	811
		25	139	31	2	635	176	1	0	583
		50	238	24	3	473	356	1	0	344
		75	385	57	3	243	482	3	0	161
		100	572	41	0	43	704	0	0	0
2	1345	0	0	0	0	793	0	0	0	795
		25	97	35	0	664	187	3	3	600
		50	192	73	9	470	348	4	4	432
		75	299	126	5	368	541	0	1	191
		100	347	162	9	222	801	0	0	0
3	3	0	0	0	0	851	0	0	0	813
		25	114	37	8	665	199	6	0	591
		50	261	52	20	536	373	5	0	413
		75	348	87	31	370	536	10	0	182
		100	467	138	41	190	797	3	0	0
3	1245	0	0	0	0	927	0	0	0	878
		25	112	20	4	755	184	0	0	606
		50	238	42	22	553	393	0	0	387
		75	337	119	11	370	628	0	0	186

		100	483	208	13	127	819	0	0	0
4	4	0	0	0	0	681	0	0	0	749
		25	93	26	2	527	169	0	0	513
		50	187	35	11	373	325	0	0	314
		75	345	61	12	263	513	0	0	139
		100	358	66	16	176	588	0	0	0
4	1235	0	0	0	0	679	0	0	0	714
		25	103	7	2	589	190	2	0	524
		50	147	15	5	501	320	2	0	388
		75	253	36	7	399	520	2	0	181
		100	295	53	2	325	716	4	0	0
5	5	0	0	0	0	728	0	0	0	688
		25	85	53	2	556	163	0	0	501
		50	183	97	2	446	352	0	0	376
		75	292	168	1	291	554	0	0	190
		100	431	240	5	76	704	0	0	0
5	1234	0	0	0	0	736	0	0	0	728
		25	36	2	23	667	42	11	31	660
		50	71	1	61	571	70	7	57	610
		75	73	1	102	536	97	0	73	598
		100	87	0	142	467	86	0	128	474

Table A.1: Number of recommendations per certainty level for initial set of experiments

Appendix B

Additional Experiment Results

Organization	Merge states with identical inflow									
	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
1	6.065	5.647	6.535	5.389	6.273	5.979	6.551	6.146	6.025	6.115
2	5.973	6.114	6.609	6.958	6.077	7.046	6.509	6.156	6.295	6.262
3	6.543	6.458	6.398	5.903	6.197	5.959	5.521	6.018	6.286	6.753
4	6.192	6.370	7.130	6.948	6.497	7.126	6.534	6.752	5.805	6.546
5	5.174	5.712	5.305	5.735	5.867	5.161	5.521	5.526	5.502	5.830

Table B.1: Avg. flow times in days of 10 repetitions of experiments with 0% recommendations

Organization	Event logs	Merge states with identical inflow									
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
1	1	5.947	6.829	6.487	6.394	6.736	6.102	5.742	6.168	6.256	5.952
1	2345	6.310	6.177	6.466	6.984	6.019	6.480	6.470	6.080	5.901	6.062
2	2	5.966	6.575	5.709	6.103	6.688	6.277	5.792	6.434	5.810	6.486
2	1345	6.091	6.359	5.680	6.322	6.302	5.729	5.298	6.392	6.581	6.738
3	3	5.859	5.725	5.275	5.621	5.407	5.458	5.978	5.639	5.528	5.522
3	1245	5.626	5.804	5.834	5.806	5.489	5.647	6.285	5.703	5.753	6.333
4	4	5.914	6.294	6.138	6.435	6.716	5.828	6.700	6.959	6.529	6.582
4	1235	6.286	6.114	6.047	6.470	7.140	6.382	6.763	6.391	6.482	6.668
5	5	5.753	5.593	5.626	6.062	5.735	5.624	5.875	5.498	4.884	5.541
5	1234	5.867	5.168	5.228	5.646	5.708	4.943	5.490	5.418	5.615	5.298

Table B.2: Avg. flow times in days of 10 repetitions of experiments with 25% recommendations

Organization	Event logs	Merge states with identical inflow									
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
1	1	6.212	6.592	6.587	6.308	6.552	5.818	6.485	6.214	6.129	6.469
1	2345	6.124	5.979	5.307	6.164	5.643	6.730	5.568	5.868	6.035	6.883
2	2	5.795	5.797	6.206	5.404	5.872	6.413	5.202	5.735	5.844	6.314
2	1345	6.766	6.336	6.501	6.366	7.166	6.514	6.100	5.638	6.765	6.620
3	3	5.106	4.868	5.105	5.326	5.684	5.022	4.732	5.303	4.906	5.382
3	1245	4.906	5.462	5.231	5.324	5.401	5.145	5.150	5.028	5.477	5.571
4	4	6.627	6.967	6.589	6.744	6.565	6.988	7.235	6.834	6.414	6.754
4	1235	6.136	6.739	6.216	6.640	6.343	6.765	6.734	6.259	6.069	6.389
5	5	5.466	5.621	5.221	5.683	5.526	5.601	5.912	5.556	5.791	5.778
5	1234	5.318	5.681	5.725	5.402	5.695	6.004	5.285	4.819	5.506	5.608

Table B.3: Avg. flow times in days of 10 repetitions of experiments with 50% recommendations

Organization	Event logs	Merge states with identical inflow									
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
1	1	6.177	6.481	5.376	6.440	6.763	6.674	6.013	6.416	6.269	6.999
1	2345	6.029	5.783	5.363	5.573	5.788	6.049	5.919	5.302	5.401	5.411
2	2	6.099	6.363	5.611	4.835	5.276	6.066	5.609	5.456	5.930	5.318
2	1345	6.563	5.866	5.993	6.850	5.784	6.525	5.803	7.377	5.207	5.682
3	3	5.082	4.826	5.062	4.455	4.597	4.853	4.693	4.571	4.472	4.732
3	1245	4.824	4.734	4.436	4.691	4.382	4.781	4.887	4.636	4.716	4.601
4	4	7.261	5.460	5.408	7.129	7.258	6.569	6.809	7.365	7.267	7.023
4	1235	6.663	7.252	6.057	6.511	6.867	5.485	6.865	6.788	6.659	5.217
5	5	5.766	5.569	6.023	5.404	6.371	5.659	6.042	6.299	5.752	5.325
5	1234	4.904	5.725	5.077	5.472	4.738	5.744	5.291	5.607	5.555	4.926

Table B.4: Avg. flow times in days of 10 repetitions of experiments with 75% recommendations

Organization	Event logs	Merge states with identical inflow									
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
1	1	5.395	5.179	4.939	5.662	4.920	4.805	5.231	4.982	5.057	4.185
1	2345	4.124	3.961	4.485	4.124	4.683	4.366	4.030	4.532	3.883	3.785
2	2	4.548	4.415	4.841	4.801	4.924	4.844	4.497	4.918	4.631	4.864
2	1345	4.521	4.856	5.159	4.594	4.958	5.265	4.920	5.213	5.080	4.842
3	3	4.135	4.182	4.048	4.341	4.038	4.145	4.138	3.965	4.138	4.103
3	1245	3.766	4.089	3.749	3.773	3.817	4.259	4.019	4.063	4.157	4.186
4	4	4.900	5.460	5.587	5.981	5.327	5.380	5.749	4.901	5.134	5.628
4	1235	5.508	4.632	5.152	5.010	5.053	4.661	5.014	5.319	5.164	5.151
5	5	5.598	5.643	6.008	5.972	5.407	6.345	5.526	5.897	6.013	6.220
5	1234	5.725	4.872	5.465	5.472	5.379	6.089	5.783	5.792	6.016	5.288

Table B.5: Avg. flow times in days of 10 repetitions of experiments with 100% recommendations

Appendix C
Approach Resource Dimension

As indicated in the future work section of Chapter 7, a possibility for further research is the effect of different resource patterns on performance. This appendix describes the steps which are already taken to be able to test that effect. Required changes to the components of the implementation, given in Chapter 5, are explained. After first describing the change to the Collective Transition System Miner, adapted executable process models which request and process recommendations about both control-flow and resources are elaborated on. Finally, extensions of the algorithms of the Multiple Log Recommender are described. Note that these implementations can serve as a first step to enable further research on the resource dimension, but are not extensively tested.

Collective Transition System Miner

Section 5.1 described how the collective transition system is created from multiple event logs. Various settings are used to determine the different states of the process model. Because in this work, the impact of control-flow configuration patterns on performance is researched, information about resources was not used when mining the transition system. However, because the impact of different resource patterns will be investigated, clearly that data has to be included in the collective transition system. As a result, the Collective Transition System Miner is changed such that it can separate states with identical control-flow, but varying resources.

Executable Process Models

The executable process model which generates an event log does not need to be adapted, since the resource already is recorded.

Compared to the executable process model that requests and processes recommendations, elaborated on in Section 5.2, a number of changes are made. Therefore, for a process instance, it will be requested what to do next and at the same time, who should do this.

Figure C.1 shows the process of requesting such a recommendation. Because a recommendation will contain information about the resource that has to carry out an activity, when requesting a recommendation, no resource should be already selected for the corresponding process instance. When no work can be carried out in the model, ‘Select Instance’ selects a case from place ‘Offered’. For each activity possible for this case, the resources capable of executing it are retrieved by the transition ‘select capable resources’. As a result, all qualified resources are

specified in the recommendation request. Furthermore, for any executed activity included in the request, it is indicated which resource performed it. Figure C.1 shows a recommendation request for the process instance with identifier '1'. Both possible activities 'receive request' and 'ask for advice' can be executed by an 'administrator' or 'trainee' resource. No activities were executed since the previous recommendation request for this case, if any.

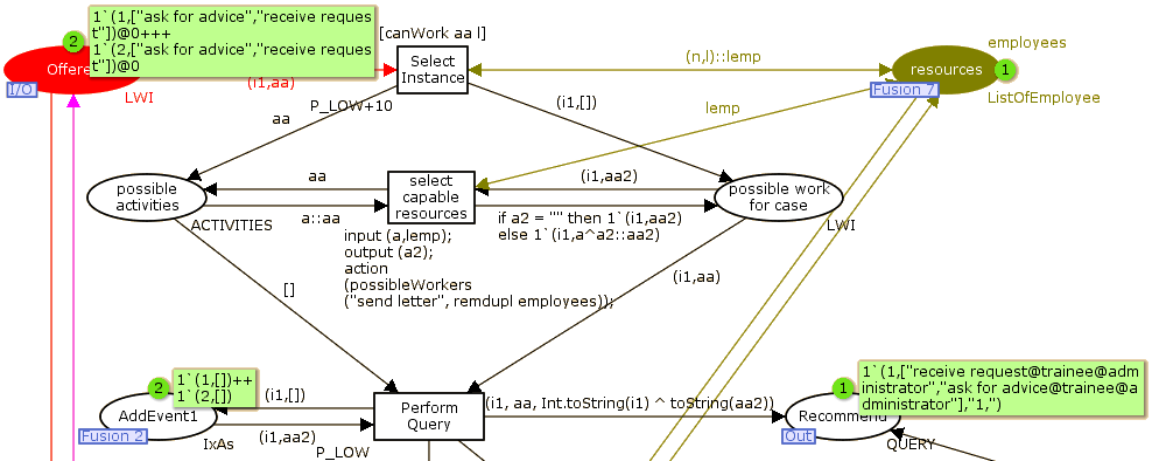

Figure C.1: The process of requesting a recommendation extended with resource information

The Multiple Log Recommender responds to such a recommendation request with which resource should carry out which activity. Figure C.2 visualizes the implementation in the executable process model that processes a recommendation. The request of the previous example is responded with '(1, "ask for advice@administrator")'. Therefore, transition 'Pick Recommended' will take a resource 'administrator' and remove the activity 'ask for advice' from place 'Offered'. The resource will start executing this activity immediately.

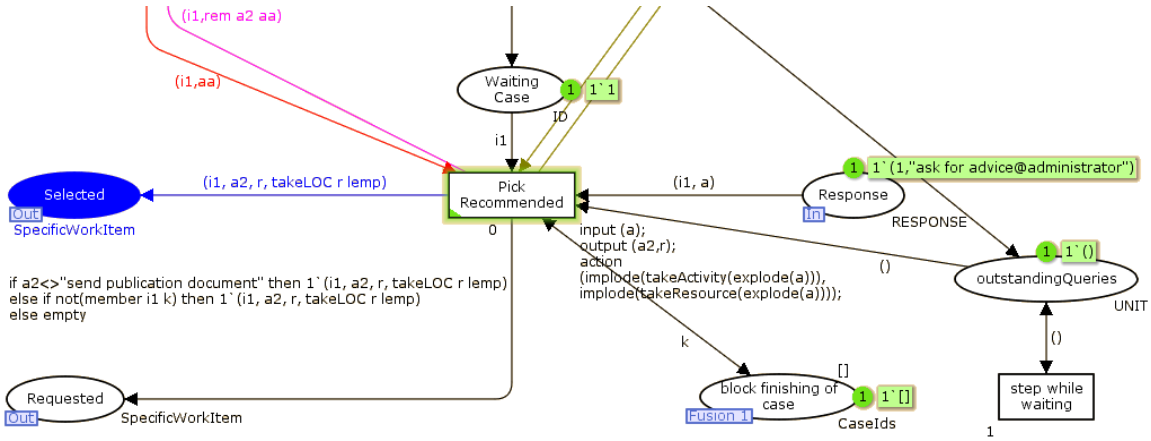

Figure C.2: The process of handling a recommendation extended with resource information

The design of this executable model allows for testing all kinds of resource patterns, such as patterns with varying number or competences of resources, duration distributions, work

schedules, policies etc. The reason for this is that these patterns can be included in the model such that they update the place ‘employees’ according to the availability of the resources.

Multiple Log Recommender

The previous paragraph showed that recommendation requests are extended with information about resources. For the executed activities it is indicated which resource performed it. For possible activities it is stated which resources are capable of executing it. Therefore, the Multiple Log Recommender is modified such that it can process this information and provide a recommendation about the next activity that should be carried out and the corresponding resource who should do this.

More specifically, the executed and possible activities used in the algorithms in Listings 5.1 – 5.3 are extended with resource information. When determining the current position candidates, transitions connecting states in the collective transition system do not only need to equal executed activity names, but also resource names. Furthermore, transitions connecting current position candidates and states need to equal possible activity and corresponding resource names when generating a recommendation.