Knowl Inf Syst (2009) 21:267–297
DOI 10.1007/s10115-009-0224-0

REGULAR PAPER

Redesigning business processes: a methodology based on simulation and process mining techniques

Laura Măruşter · Nick R. T. P. van Beest

Received: 15 October 2008 / Revised: 27 March 2009 / Accepted: 17 May 2009 /
Published online: 25 June 2009

Abstract Nowadays, organizations have to adjust their business processes along with the changing environment in order to maintain a competitive advantage. Changing a part of the system to support the business process implies changing the entire system, which leads to complex redesign activities. In this paper, a *bottom-up* process mining and simulation-based methodology is proposed to be employed in redesign activities. The methodology starts with identifying relevant performance issues, which are used as basis for redesign. A process model is "mined" and simulated as a representation of the existing situation, followed by the simulation of the redesigned process model as prediction of the future scenario. Finally, the performance criteria of the current business process model and the redesigned business process model are compared such that the potential performance gains of the redesign can be predicted. We illustrate the methodology with three case studies from three different domains: gas industry, government institution and agriculture.

Keywords Process redesigning · Process mining · Simulation · CPN Tools

1 Introduction

Current business environments evolve increasingly fast due to new markets, changing government rules and emerging technologies. As a result, organisations have to adjust their business processes along with the changing environment in order to maintain a competitive advantage [11,19]. Furthermore, business processes may not perform as desired according to various measures, and have to be redesigned consequently. The causes of these performance issues should be identified in order to propose suitable redesign goals that may solve the

L. Măruşter (✉) · N. R. T. P. van Beest
Department of Business and ICT, Faculty of Economics and Business,
University of Groningen, PO Box 800, 9700 AV Groningen, The Netherlands
e-mail: l.maruster@rug.nl

N. R. T. P. van Beest
e-mail: n.r.t.p.van.beest@rug.nl

performance issues of the business process. Additionally, the appropriate indicators should be formulated, in order to be able to verify whether the initial goals were met and determine the success of the redesign.

There is a vast literature concerning redesign of business processes [6,11,13] and simulation of business processes [4,16]. Redesign of a business process can be executed based on process documentation (top-down) or based on process data (bottom-up). The latter approach focusses on identifying performance issues of the actual process and potential improvements based on available process data. Process models can be obtained using automatic discovery of information from event logs, which is called process mining. The discovered models can be used as a feedback tool that helps auditing, analyzing and improving already enacted business processes. For an extensive overview about process mining, see [23].

There have been attempts to combine process mining and simulation techniques in order to perform simulations of the business process using the obtained process models as a starting point. For instance, Rozinat et al. refer in [15] to the Protos2CPN tool [5], which translates Protos process models into Coloured Petri nets (CP-nets or CPNs). Coloured Petri nets combine the strengths of ordinary Petri nets with the strengths of a high-level programming language [2,8]. Rozinat et al. [15] show that a business process can be discovered and then simulated with the aid of the ProM framework [21] and CPN Tools. The authors have illustrated this approach with one artificial example and two real cases. They also mention that "for structured processes this works well. However, for more chaotic processes it is difficult to produce models that are easy to interpret and analyse". Furthermore, the case studies discussed in Rozinat et al. [14] seem to be very similar to each other: the institutions considered were both municipalities, whereas the information systems producing the logs were workflow systems.

This paper proposes a methodology that combines process mining with simulation, which enables an organisation to redesign the business process and predict the future performance of that process based on the simulation. However, in contrast to existing approaches, the proposed methodology is explicitly addressing less structured processes, which are supported by various (legacy) information systems.

In Fig. 1, the proposed approach is presented, where three process models are constructed: (i) model M1 based on the logged data, (ii) model M2 that aims to simulate the current As-Is situation and (iii) model M3 that proposes a redesign (To-Be) scenario. The performance criteria corresponding to each of these three models (P1, P2 and P3) are reported for comparison purpose. In this paper, three distinct case studies are presented to explicitly describe the application of this methodology in practice, and to illustrate its potential results. In addition, for each case study, an example will be provided on how the redesign scenarios can be interpreted. The case studies have been specifically selected to be distinct, in order to illustrate how to deal with specific difficulties encountered in each case.

The article is organised as follows. In Sect. 2, the three case studies are described and investigated with respect to the performance criteria issue. The mining of the As-Is process model and related substeps are presented in Sect. 3. The simulation of process models in the As-Is and the To-Be situations are shown in Sect. 4. The last step of the proposed methodology, concerning comparison of the As-Is with the To-Be models, is presented in Sect. 5, followed by the conclusions in Sect. 6.

2 Case studies and identification of performance criteria

Redesigning business processes focusses on specific redesign objectives, which can be expressed by selecting the appropriate performance criteria. The performance criteria define

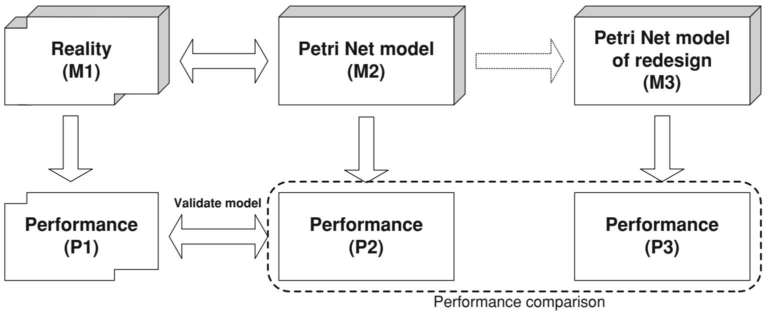


Fig. 1 The overall approach

how the business process is assessed. Based on this performance assessment, the proposed redesign will be created.

Redesign goals can range from incremental to radical change. According to Hammer and Champy [6] in Business Process Re-engineering (BPR) the central topic is improving the management of processes. According to Davenport and Short [3], "BPR is driven by a business vision which implies specific business objectives such as Cost Reduction, Time Reduction, Output Quality Improvement, Quality of Work Life, Learning and Empowerment. For Johansson [9], BPR is used to improve the performance of organisations as measured by cost, cycle time, service and quality. Short and Venkatraman [17] emphasised on the customer point of view as the BPR goal, namely the focus on improving product distribution and delivery performance to the customer.

The focus of BPR may concern different aspects: product standardisation, tasks design, routing, prioritising, information exchange, work allocation, IT support, process information, production control, and management style [25]. Consequently, Key Performance Indicators (KPIs) such as service time (total time spent on case), waiting time (total idle time of the case), resource utilisation, number of errors, etc., must be measured [25].

Depending on the business objectives and the redesign goal, adequate performance criteria need to be considered. In order to illustrate our proposed methodology, we selected three different case studies: a business process in a Dutch company active in the gas sector, the workflow process of a Dutch governmental organisation and a decision support system supporting the decision making activities of a Dutch agricultural organisation. These case studies are different with respect to their business goals and redesign objective, and, therefore, present different challenges to the proposed methodology.

2.1 Case study 1: Gas company

The first case study considered is a Dutch company[1] from the gas sector; we refer to the company as the Gas company. The liberalisation of the gas market has led to the emergence of challenges such as new and different market players, new business processes, new roles and new information needs. In response to these changes, the Gas company has to redesign its business processes, including its information systems and the underlying processes. One of the changes from the gas sector after liberalisation is the increased externalisation. Information needs from external parties such as shippers (the trading companies in the gas market) become as important as internal information needs, which both impact existing IT architectures and

[1] The names of the institutions considered in this paper are omitted because of confidentiality reasons.

infrastructure. Furthermore, as the number of customers and their demands increase, performance is becoming an important issue. Therefore, the Gas company wants to redesign its business processes and the information systems supporting these processes, while achieving high-performance gains.

In case of the Gas company, the focus is to redesign, among others, one of the processes involved in booking gas capacity, considering some operational performance measures. The main concern is the entire throughput time of this process, as customers want to be able to get the service as fast as possible. *Throughput time* is defined as the average time that is required to perform an activity or a process [7]. A *bottleneck* is defined as a step in a process that has the highest utilization of the entire process [7]. High utilization easily results in high throughput times caused by long queueing times [7]. For this reason, performance can be influenced by the location and severity of the bottlenecks present in the process. Therefore, the number of bottlenecks and their severity are also used as performance measures.

2.2 Case study 2: Governmental institution

The second case study is a Dutch governmental institution responsible for collecting fines. The process of fine collection concerns the entire flow of activities from initial regulation to payment, which includes the first and second reminders.

Over the last few years, the work of the governmental institution has grown to the nine main activities currently being executed. As a result of this gradual growth, and due to specific problems of these main activities, for each of them a bespoke information system was developed. Subsequently, the increasing number of systems became more difficult to manage over the years. Furthermore, the change of the current systems, as well as the development of support for new products became more time-consuming. At the moment when this case study was performed, the current systems have been developed to process approximately 3 million cases, when actually the need was to process nearly 10 million cases per year.

The Governmental institution was interested to investigate the process underlying the workflow of processing fines with the aim of rearchitecting the IT infrastructure. As a result of the increasing amount of cases, the IT capacity was insufficient. Therefore, the governmental institution was interested in performance measurements (such as throughput time and number of bottlenecks), in order to redesign the IT infrastructure in case of structural problems (deadlocks, infinite loops, etc.). The company intended to reorganise their processes in order to increase efficiency and decrease throughput time.

A *case* (process instance) is a fine that has to be paid. Each fine corresponds to one independent case. The particularity of this process is that it stops, as soon as the fine is paid. In total there are 99 distinct activities, denoted by numbers, which can be executed either manually or automatically. The fines information is collected for 130,136 cases.

2.3 Case study 3: decision support system

The third case is about redesigning a Web-based decision support system (DSS) that supports agricultural users. A DSS is perceived as a computer system that aids people in making a decision regarding a specific domain [10]. The goal of this specific DSS is to ensure that agricultural users gain insight into damages caused by some parasites and the financial consequences of the selected cultivar. However, it was shown that the DSS was not used by the agricultural users as planned [12]. In contrast to Shyu et al. [18], the goal of this case is not only to discover usage patterns, but also primarily to identify redesign incentives to improve the fulfillment of the Web site goal.

The interface of DSS consists of several pages. Web pages do not have the same function, which depends on the purpose of the Web site. In our case, since the goal of the site is to provide the agricultural users with information about different yield scenarios given the chosen factors, the following types of pages can be distinguished: (i) the pages reflecting the site's goal, e.g. containing information about the yield, (ii) the pages that can lead to fulfilling the site's goal, and (iii) the other pages. Using the notion of *concept hierarchies*, or *service-based hierarchies* [20], we determine *action pages* (a page whose invocation indicates that the user is pursuing the site's goal), and *target pages* (a page whose invocation indicates that the user has achieved the site's goal).

According to the mapping between Web pages and phases of the decision making process mentioned in Mărușter et al. [12], 'Action' pages correspond to the *Information gathering* phase (where the user is supported to explore the selection of the cultivar, by specifying field characteristics and the PCN history), and 'Target' pages correspond to the *Design* phase (where the system provides the user with one or more possible solutions). The *Choice* phase does not have any corresponding pages in the DSS, the final decision is actually taken by the user outside of the system. Based on this mapping, the DSS is said to support the *Information gathering* and *Design* decision making phases if the corresponding 'Action' and 'Target' pages are visited to a comparable extent. This statement is used to establish performance criteria for redesign, e.g. goal fulfillment. Redesign measures have to be implemented when the statement is not satisfied.

3 Mining the As-Is process model

Most of the organisations use information systems to support the execution of their business processes. These information systems typically support logging capabilities that register the tasks or activities which have been performed in the organisation. Typical log data usually consist of cases (i.e. process instances) that have been executed, the times at which the tasks were executed, the persons or systems that performed these tasks and other kinds of data. Such logs, known as *event logs*, are the starting point for process mining.

The process mining methods target the automatic discovery of information from such an event log [23]. This discovered information can be used to deploy new systems that support the execution of business processes or as a feedback tool that helps in auditing, analysing and improving already enacted business processes. Since process mining techniques gather information about the actual flow instead of the alleged process flow, some differences may be expected between the processes identified by the logs and the formal description of the process as provided by available documentation.

3.1 Logging and data processing

The data used in process mining may or may not require further processing steps, depending on existing system log functionalities, logging objectives, etc. The three cases considered in this paper illustrate three different situations:

1. Gas company case: the logging functions have been designed with a different purpose than process mining. Therefore, additional conversion steps are necessary to process the log data in order to prepare them for process mining;
2. Governmental institution: no conversion was necessary;
3. DSS case: additional functions have been embedded in the application to make process mining possible.

In case of the Gas company, three types of logs were available, depending on the type of information systems: (i) the log of the ERP system, (ii) the log resulting from the interaction between the front-end system and company's clients, and (iii) the log of interfacing between these two systems. The logged data consisted of a number of irrelevant fields and details, and had to be converted to a suitable format before they could be used in process mining.

The analysis starts with identifying the complexity of the log data. First, the original log data belonging to the considered process consist of 142 different activities, which occur in various process instances. Second, the level of aggregation in the raw log data showed strong variations. For process mining, it is essential that activities recorded in the logs share the same level of aggregation, in order to be comparable and consistent. The eventlog data as provided by the information system may be very inconsistent concerning the level of granularity, because in many cases the application developer decides ad hoc to log a certain activity during development time. For instance, the log contains high-level activity entries like for example 'Booking', and low-level entries such as 'Registry error'.

Determining the level of granularity and grouping is not a feature that is provided by ProM, as this is very context dependent. Therefore, it should be performed in consultation with a process expert of the case at stake. In order to construct a suitable log file, two basic types of conversions should be executed. First, the raw data are merged in a log file containing data at the same aggregation level. Second, the obtained data are converted into a data format acceptable by the ProM framework.

An example of the raw data is shown in Fig. 2. This raw data log file originates from three different log types: the on-line application providing Front-end customer support, the ERP system and the Interface application. When investigating the raw logdata, related activities, which we will call low-level activities, can together constitute a higher level activity. The higher level activity is represented by registering the start and end time stamp, which corresponds to the timestamp of the first low-level related activity, and the timestamp of the last low-level related activity, respectively. As a result, the group of low-level activities is replaced by one single high-level activity entry comprising two rows, stating the start and end timestamp. In this case, the timestamps of the first low-level activity of the group is used as start event, and the timestamp of the last low-level activity of the group is used as end event. For instance, in Fig. 2, the first three low-level activities 'Booking inserted', 'WorkFlow DoNextstep', and 'Bookingstatus = 12' could be merged into a high-level activity 'Request For Booking'.

Determining the suitable level of aggregation throughout the log files it is not straightforward. Therefore, consultations with the process expert are required in order to determine the appropriate level of aggregation. The identification of the start and end of a high-level activity can be done in the following ways:

- Based on activity names. Activities may share common labels in their names. Corroborating with the fact that such activities occur in a particular order, it may be a clue that they could be merged. For identifying the criteria of determining high-level activities based on activity names, a process expert should be consulted, who has a better understanding of all activities. For instance, in Fig. 2, activities related with "financial check" can be merged into one high-level activity "Financial Check".
- Based on user type. The activities performed by certain user types can be merged as one high-level activity. For instance, in Fig. 2, the first five activities performed by the front-end users ('User_id') can be merged into one high-level activity.

ID	TimeStamp	Rec_id	Log_id	User_id	Order_id	Activity	Logtype	Code	Eventtype
1	1-12-2006 9:20:01	1	200000	FrontEnd	1000	Booking inserted	FrontEnd	USER	Complete
1	1-12-2006 9:20:02	2	200001	FrontEnd	1000	WorkFlow DoNextStep	FrontEnd	USER	Complete
1	1-12-2006 9:20:02	4	200002	FrontEnd	1000	Bookingstatus = 12	FrontEnd	USER	Complete
1	1-12-2006 9:20:02	6	200004	FrontEnd	1000	Request saved to be sent	Interface	USER	Complete
1	1-12-2006 9:20:04	9	200005	FrontEnd	1000	Request sent to ERP: OK	Interface	USER	Complete
1	1-12-2006 9:20:06	12504	100	ERP	1000	Check: request type booking	ERP	USER	Complete
1	1-12-2006 9:20:07	12505	200	ERP	1000	Check: validate incoming data	ERP	USER	Complete
1	1-12-2006 9:20:07	12506	250	ERP	1000	Check: result: validity check	ERP	USER	Complete
1	1-12-2006 9:20:08	12507	300	ERP	1000	Check: technical check needed	ERP	USER	Complete
1	1-12-2006 9:20:24	12508	350	ERP	1000	Perform: technical check	ERP	USER	Complete
1	1-12-2006 9:20:24	12509	400	ERP	1000	Check: interruptible check needed	ERP	USER	Complete
1	1-12-2006 9:20:24	12510	450	ERP	1000	Check result: technical check	ERP	USER	Complete
1	1-12-2006 9:20:24	12511	600	ERP	1000	Check: financial check needed	ERP	USER	Complete
1	1-12-2006 9:20:25	12512	650	ERP	1000	Perform: price calculation	ERP	USER	Complete
1	1-12-2006 9:20:25	12513	700	ERP	1000	Perform: financial check	ERP	USER	Complete
1	1-12-2006 9:20:25	12514	900	ERP	1000	Interface: send FrontEnd reply	ERP	USER	Complete
1	1-12-2006 9:20:31	10	200008	ERP	1000	FrontEnd ERP reply	Interface	USER	Complete
1	1-12-2006 9:20:31	11	200009	FrontEnd	1000	WorkFlow DoNextStep	FrontEnd	USER	Complete

Fig. 2 Example of the Gas company raw log file

– Based on logtype change. The logtype may indicate, for example, which system generated a particular activity. The 'Logtype' attribute in Fig. 3 shows that an activity can be performed by the FrontEnd system, by the ERP system, or the Interface system.

We choose to merge the raw data based on the logtype change. Using the attributes TimeStamp and Rec_id, the first three activities 'Booking inserted', 'Workflow DoNextStep' and 'Bookingstatus = 12' are merged into a new activity called 'Request For Booking'. The results of the merging method are shown in Fig. 3. Note that in the merged log file, the column 'Eventtype' indicates the start and the end of that high-level activity 'Request For Booking'. The log-entries generated by the Interface system ('LogType' Interface) are reproduced without changes, because we are also interested in tracking throughput times in between the on-line application and ERP system.

In the situation of the Governmental institution, a *case* (process instance) is a fine that has to be paid. Each fine corresponds to an independent case. In total there are 99 distinct activities, denoted by numbers, which can be executed either manually or automatically. The log is the result of recording activities of a workflow system, which has built-in functionality for logging each workflow step.

Unlike the Gas company case, the data of the Governmental institution did not require any merging, due to the consistent level of aggregation. The activities in the log file correspond to a certain process-step in the workflow-system, resulting in a comparable level of granularity. However, because of the high amount of activities, only those activities that occur in more than 2% of all activities in a case will be considered in further analysis. The total amount of data relates to 130,136 cases. In order to speed up the analysis process, a random sample of 5% of the process instances has been selected. An example of the raw data of the Governmental institution[2] is shown in Fig. 4. The Planned_date and Executed_date were used as timestamps for the beginning and the end of an activity (Trigger_date attribute was ignored).

Our third case study, which concerns a DSS application, contained Java Beans built-in functionality for logging navigation behaviour. Agricultural user's navigation behaviour is logged in a *movement file* (MOV), which contains information concerning the movement from one 'source' page to a 'destination' page, such as time stamp, name of source page, name of destination page, and the button used. Table 1 shows an example of the DSS navigation sequence, recorded in a MOV file. It can be seen that the user sequentially accesses five types of pages, mostly by pressing the 'Next' button. In total, 763 user sessions belonging to 501 individual users have been logged and employed in the analysis.

Finally, for all case studies, the raw data are converted with the aid of the ProM*Import* framework to a format that is accepted by the process mining tool ProM [21].

3.2 Obtaining the "reality" process model

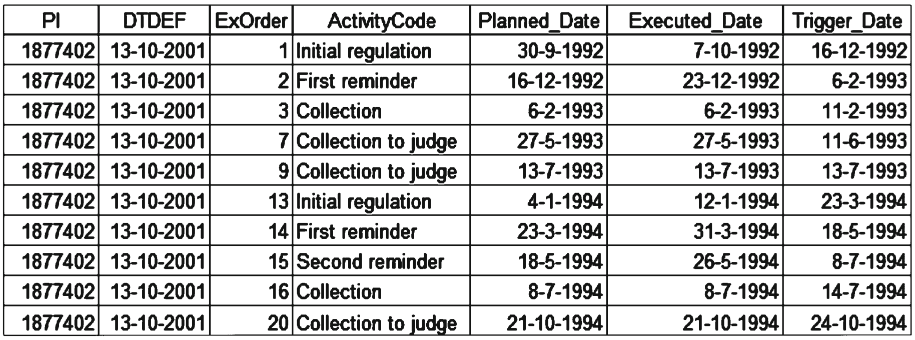
A process model can be constructed with the aid of process mining tools using different sources of information. The result of the process mining could have different representations: heuristics nets, Petri nets, Event Process Chains (EPC), etc.

In order to construct a process model, a suitable process mining algorithm should be selected, which is able to produce a Petri net. However, because the quality of the log data is uncertain, we need an algorithm robust to noise and exceptions. The heuristic miner algorithm seems to be a good candidate, since it is based on patterns frequencies, which makes it

[2] For confidentiality reasons, the labels of original data are translated from Dutch to English, according to the own interpretation of the authors.

ID	TimeStamp	Rec_id	Log_id	User_id	Order_id	Activity	Logtype	Code	Eventtype
1	1-12-2006 9:20:01	1	0	FrontEnd	1000	Request For Booking	FrontEnd	USER	Start
1	1-12-2006 9:20:02	2	0	FrontEnd	1000	Request For Booking	FrontEnd	USER	Complete
1	1-12-2006 9:20:02	3	200004	FrontEnd	1000	Request saved to be sent	Interface	USER	Complete
1	1-12-2006 9:20:04	4	200005	FrontEnd	1000	Request sent to ERP: OK	Interface	USER	Complete
1	1-12-2006 9:20:06	5	100	ERP	1000	MiniWorkflow	ERP	USER	Start
1	1-12-2006 9:20:25	6	900	ERP	1000	MiniWorkflow	ERP	USER	Complete
1	1-12-2006 9:20:31	7	200008	ERP	1000	FrontEnd ERP reply	Interface	USER	Complete
1	1-12-2006 9:20:31	8	200009	FrontEnd	1000	Request For Booking	FrontEnd	USER	Start
1	1-12-2006 9:20:31	9	200009	FrontEnd	1000	Request For Booking	FrontEnd	USER	Complete

Fig. 3 Example of merged Gas company log file



PI	DTDEF	ExOrder	ActivityCode	Planned_Date	Executed_Date	Trigger_Date
1877402	13-10-2001	1	Initial regulation	30-9-1992	7-10-1992	16-12-1992
1877402	13-10-2001	2	First reminder	16-12-1992	23-12-1992	6-2-1993
1877402	13-10-2001	3	Collection	6-2-1993	6-2-1993	11-2-1993
1877402	13-10-2001	7	Collection to judge	27-5-1993	27-5-1993	11-6-1993
1877402	13-10-2001	9	Collection to judge	13-7-1993	13-7-1993	13-7-1993
1877402	13-10-2001	13	Initial regulation	4-1-1994	12-1-1994	23-3-1994
1877402	13-10-2001	14	First reminder	23-3-1994	31-3-1994	18-5-1994
1877402	13-10-2001	15	Second reminder	18-5-1994	26-5-1994	8-7-1994
1877402	13-10-2001	16	Collection	8-7-1994	8-7-1994	14-7-1994
1877402	13-10-2001	20	Collection to judge	21-10-1994	21-10-1994	24-10-1994

Fig. 4 Example of the raw log file in the Governmental institution case

Table 1 An example of a movement log file in the DSS case

Time stamp	FromPage	ToPage	Button
2004-12-22 22:13:29	pageA1	pageA2	NEXT_BUTTON
2004-12-22 22:13:35	pageA2	pageA3	NEXT_BUTTON
2004-12-22 22:14:00	pageA3	pageT1	NEXT_BUTTON
2004-12-22 22:14:26	pageT1	pageT2	NEXT_BUTTON
2004-12-22 22:16:16	pageT2	pageT1	BACK_BUTTON
2004-12-22 22:16:25	pageT1	pageT2	NEXT_BUTTON

is possible to focus on the main behaviour in the event log [24,26]. In addition, the heuristic net can be converted into a Petri net for performance analysis (see Sect. 3.3).

A *heuristic net* (HN) is a graphical representation, consisting of rectangles, which correspond to transitions or activities, and arcs, which indicate the existence of relationships between activities [27]. There are two special activity names, ArtificialStartTask and ArtificialEndTask that refer to a generic start or end activity indicating the start and end of the entire process. The ArtificialStartTask and ArtificialEndTask are required in any case.

The process mining tool ProM provides the ability to check the quality of the resulting process models. This is a form of conformance testing between the model and the transaction logs. A useful measure in this sense is called 'Continuous semantics fitness' (CSF), which calculates the degree with which the model is able to describe the "reality" contained in the log data. It ranges between a minimum of 0 (no fit) and maximum 1 (perfect fit) (for details see [15]).

For all the case studies, we will first use the heuristic miner algorithm for developing the process model with the default parameters. Second, we will perform the conformance check and calculate the CSF measure. If an insufficient value is obtained for this measure, subsequent process models will be constructed by adjusting heuristic miner parameters, until the CSF measure attains a sufficient value.

3.2.1 The process model of the Gas company

During the placement of an order by the customer, there is an exchange of messages between the ERP system and the system designed to support the on-line activities of the Gas company. We consider the order identifier as the process mining case.

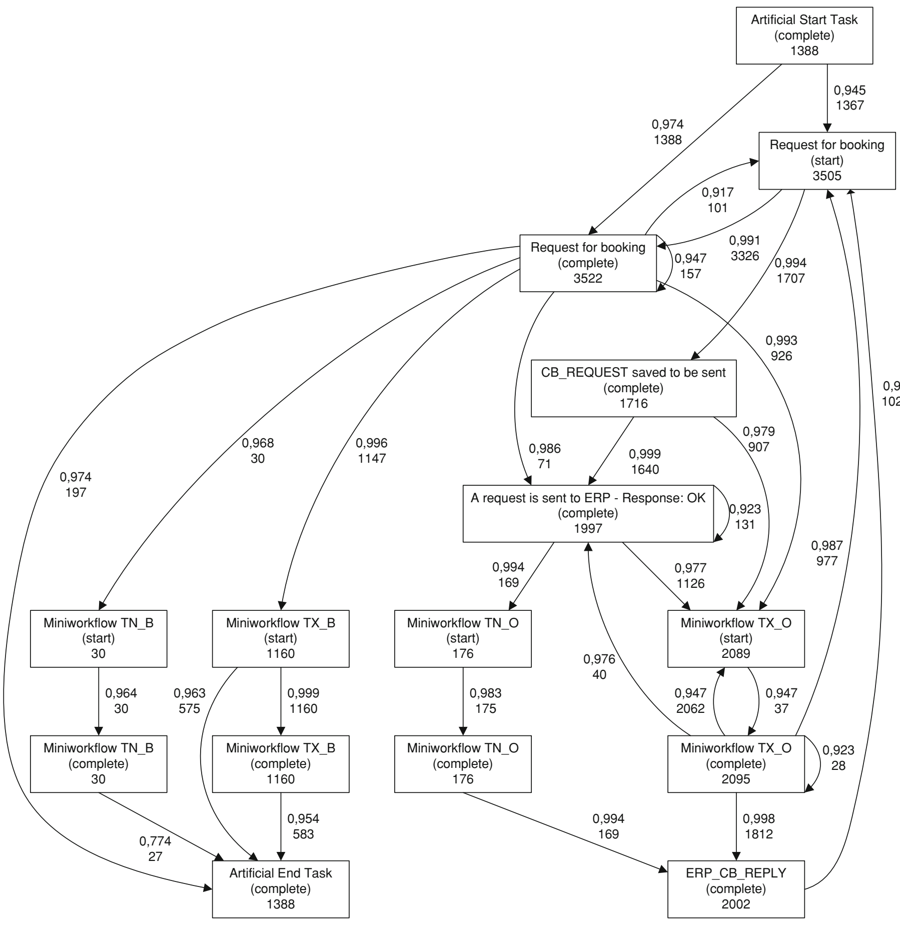

Artificial Start Task (complete) 1388
0,945 1367
0,974 1388
Request for booking (start) 3505
0,917 101
0,991 3326
0,994 1707
Request for booking (complete) 3522
0,947 157
0,993 926
CB_REQUEST saved to be sent (complete) 1716
0,98 1020
0,979 907
0,999 1640
0,986 71
0,968 30
0,996 1147
0,974 197
A request is sent to ERP - Response: OK (complete) 1997
0,923 131
0,987 977
0,994 169
0,977 1126
Miniworkflow TN_B (start) 30
Miniworkflow TX_B (start) 1160
Miniworkflow TN_O (start) 176
Miniworkflow TX_O (start) 2089
0,976 40
0,964 30
0,963 575
0,999 1160
0,983 175
0,947 2062
0,947 37
Miniworkflow TN_B (complete) 30
Miniworkflow TX_B (complete) 1160
Miniworkflow TN_O (complete) 176
Miniworkflow TX_O (complete) 2095
0,923 28
0,954 583
0,774 27
0,994 169
0,998 1812
Artificial End Task (complete) 1388
ERP_CB_REPLY (complete) 2002

Fig. 5 The process model of the Gas company represented as a heuristic net, using default parameter settings

In Fig. 5, the process model of the Gas company is represented as a heuristic net using default parameters. The term ‘complete’ refers to a finished action (in terms of a finite-state machine, an activity can be in the states ‘new’, ‘sent’, ‘active’, ‘suspended’, ‘complete’, etc.).

The number inside the rectangle shows how many times an activity has been executed. For instance, activity ‘Request for Booking (start)’ occurred 3,505 times. The arcs between two activities A and B are labelled with two numbers:[3] (i) the first number, between 0 and 1, represents the causality/dependency measure between A and B. (ii) An integer number, which represents the co-occurrence frequency of A and B. The higher the causality measure is, the stronger the relation between A and B. In Fig. 5, between activity ‘Request for Booking (complete)’ and ‘Miniworkflow TX_O (start)’ the first number 0.993 is the causality/dependency measure, and the second number 926 represents the co-occurrence frequency.

Because the Heuristic Miner is based on the frequency of patterns, it is possible to focus on the main behaviour in the event log. However, if a log file contains a large number of outliers, this may be reflected in some irregular transitions in the heuristic net. In some cases

[3] For more details, see [27].

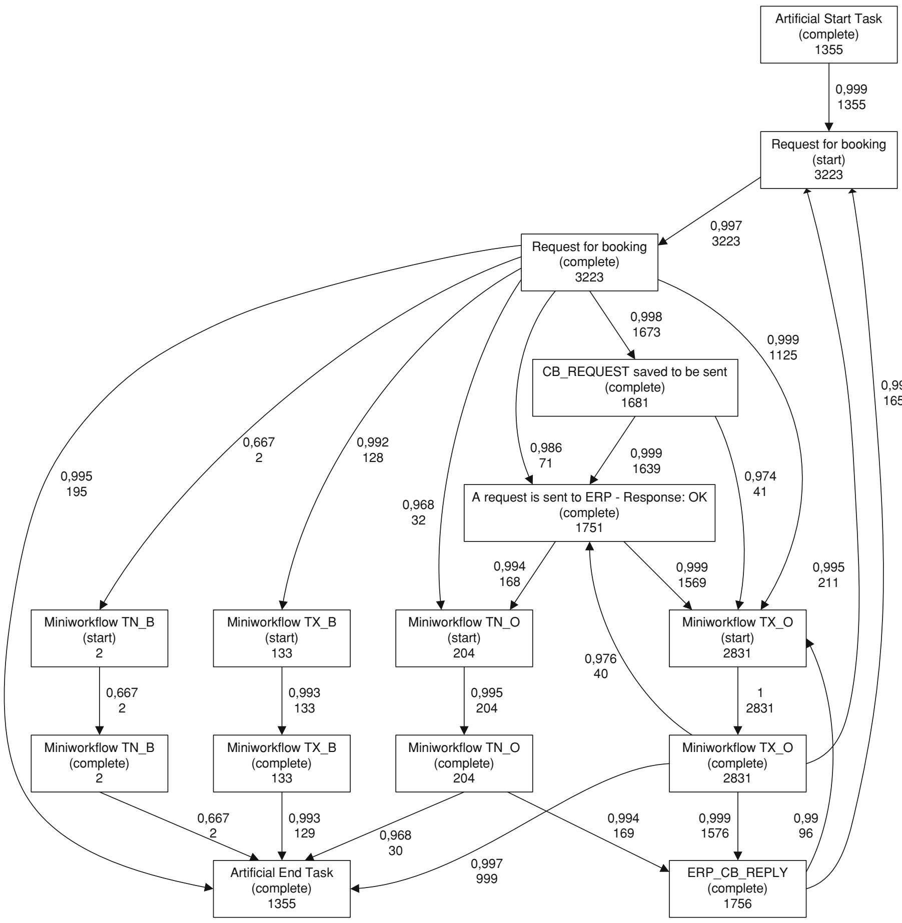

Artificial Start Task (complete) 1355
0,999 1355
Request for booking (start) 3223
0,997 3223
Request for booking (complete) 3223
0,998 1673
0,999 1125
CB_REQUEST saved to be sent (complete) 1681
0,998 1654
0,667 2
0,992 128
0,995 195
0,986 71
0,999 1639
0,974 41
0,968 32
A request is sent to ERP - Response: OK (complete) 1751
0,994 168
0,999 1569
0,995 211
Miniworkflow TN_B (start) 2
Miniworkflow TX_B (start) 133
Miniworkflow TN_O (start) 204
Miniworkflow TX_O (start) 2831
0,976 40
0,667 2
0,993 133
0,995 204
1 2831
Miniworkflow TN_B (complete) 2
Miniworkflow TX_B (complete) 133
Miniworkflow TN_O (complete) 204
Miniworkflow TX_O (complete) 2831
0,667 2
0,993 129
0,968 30
0,994 169
0,999 1576
0,99 96
0,997 999
Artificial End Task (complete) 1355
ERP_CB_REPLY (complete) 1756

Fig. 6 The process model of the Gas company represented as a heuristic net, after cleaning

these irregular transitions are obvious, like the self-loop of the *TX_O complete event* or the transition between *TX_B start* to *ArtificialEndTask*. In most of the cases, however, a process expert is required to reveal these irregular transitions in the model.

For the Gas company, the continuous semantics fitness is 0.33, which shows a moderate to low fitness between the model and the log. This could be attributed to the presence of some noisy cases or outliers. In order to resolve the influence of outliers, ProM provides a plugin to identify these irregular process instances. The Fuzzy miner plugin can be used to identify cases that have low conformance with the model. All process instances are shown, and for any given case, it is stated whether the corresponding flow of activities concern an irregular transition or not. Process instances containing 75% or more irregular transitions were eliminated.

In Fig. 6, the cleaned process model of the Gas company is shown as a heuristic net. Shippers can book transport capacity at entry points and at exit points using the front-end system. This is represented by the *Request for booking* activity. However, before awarding transport capacity to a shipper, an availability and a technical transport tests are performed. These tests are performed using the ERP system. Furthermore, the contracting and billing is handled by the ERP system. The process of both the transport tests and the contracting and billing are merged to the activity *Miniworkflow*. A distinction is made between automatic and manual activities, namely between automatic online booked capacity (TX_O and TN_O) and background processed manual bookings (TX_B and TN_B). The activities representing automatic activities are preceded and followed by interface activities, namely the exchange of messages between the front-end and the ERP system.

Based on the heuristic net model, "the most common pattern" can be obtained, which consists of *ArtificialStartTask* $\rightarrow$ *Request for booking* (1,355), *Request for booking* $\rightarrow$ *CB_Request saved to be sent* (1,673), *CB_Request saved to be sent* $\rightarrow$ *A request is sent to ERP* (1,639), *A request is sent to ERP* $\rightarrow$ *Miniworkflow TX_O* (1,569), *Miniworkflow TX_O* $\rightarrow$ *ERP_CB_REPLY* (1,576), *ERP_CB_REPLY* $\rightarrow$ *Request for booking* (1,654) or *ERP_CB_REPLY* $\rightarrow$ *ArtificialEndTask* (999).

Upon inspection of the resulting heuristic net, it can be concluded that a loop occurs before the process ends. This loop either points to the execution of the different Miniworkflow activities again, or to restarting the process with *Request for booking*. Out of the 2,831 cases, only 999 completed, while the majority of them restarted the process. The execution of activities involved in such a loop may provide interesting insights into the actual process flow of the Gas company. For example, after *ERP_CB_Reply* is executed, the *Miniworkflow TX_O* is recalled again, due to a system error that may have occurred (in which case the user should provide a revision of data). In cases where it is not possible to book capacity automatically, the orders are processed manually, which results in a switch between automatic *Miniworkflow TX_O* to *Miniworkflow TX_B*. If the order is completed or abandoned, the process ends with *ArtificialEndTask*.

After cleaning the log file, the model was tested for conformance. The 'Continuous Semantics fitness' measure is equal to 0.9895, which indicates that the conformance of this process model with the cleaned log is considerably higher than the previous process model based on uncleaned logfile.

3.2.2 The process model in case of Governmental institution

The log data from the Governmental institution contained a very large number of cases (130,136). Therefore, a random sample of 5% of the process instances was selected. Figure 7 shows a heuristic net representation of fine collection process model using default parameters.

The heuristic net of the Governmental institution also shows some irregular transitions, as represented by the self-loops of e.g. "07 (complete)". Furthermore, the low value (0.42) of CSF also implies the presence of noisy instances that had to be eliminated. Using the ProM Fuzzy miner method, outliers are identified. However, removing these outliers did not significantly increase the 'Continuous Semantics fitness' measure in this case. Improving the fit between the heuristic net and the data can be obtained by adjusting the threshold parameters for the heuristic miner. The threshold parameters indicate the conditions under which additional connections can be displayed in the model. For example, the *positive observations* parameter indicates the amount of positive observations required to create a connection between two activities in the Heuristic net. Increasing the *positive observations* threshold results in a more generic process flow. By setting this value to 200, the 'Continuous semantics fitness' increases to 0.97, which shows a very good fit between the model and log. The value of *positive observations* threshold can be repeatedly increased, until a satisfying value for the CSF measure is obtained.

In Fig. 8, the process model of the Governmental institution is shown as a heuristic net, with parameter tuning. After receiving the case-related documents, the process starts with

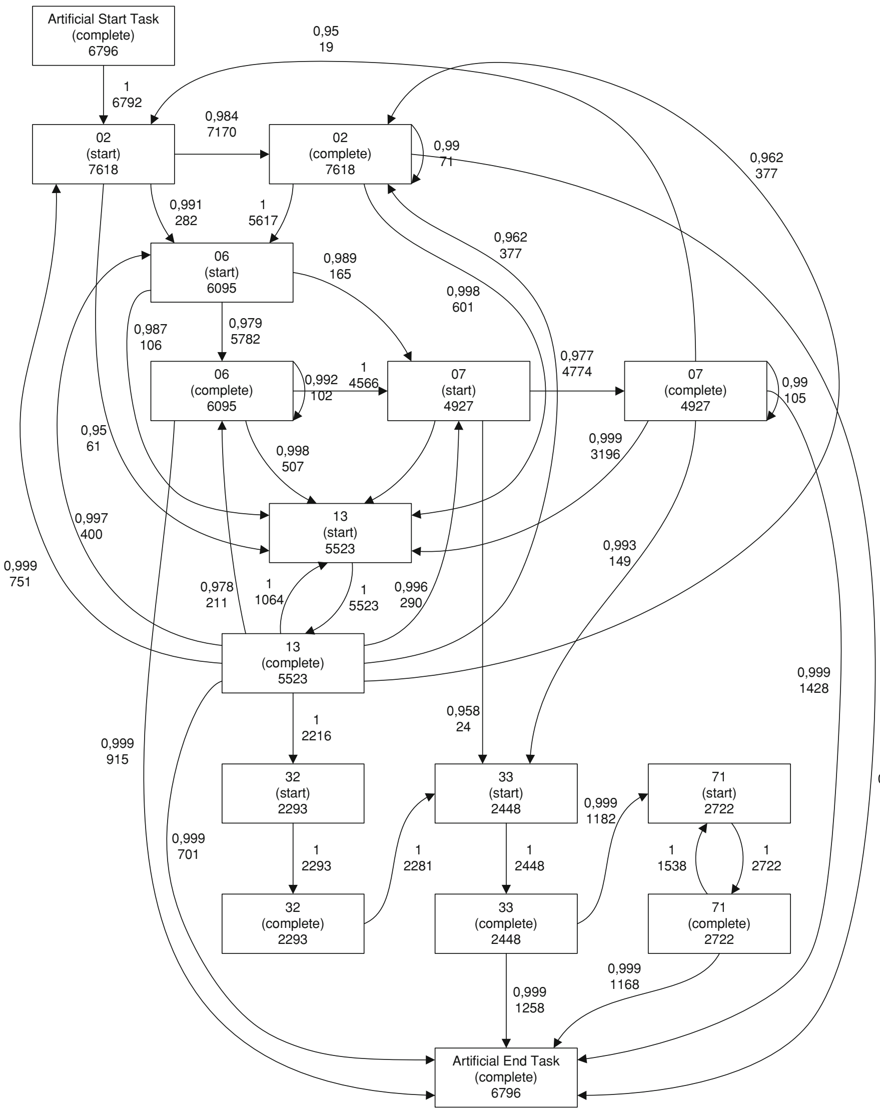


Fig. 7 The process model of collecting fines in case of the Governmental institution, represented as a heuristic net, using default parameter settings

the automated activity *Initial regulation*, which is identified with label “02”. An example of this activity could be the sending of a transfer form, specifying the fine amount that must be paid). If after 8 weeks the fine is not paid, a reminder is sent automatically via the activity “06”—*First reminder*). If still no payments are made after 8 weeks, activity “07”—*Second reminder* sends a last reminder.

After this, a manual activity *Judge standard verification* (activity “13”) follows. Note that after any of these activities a payment can follow, in which case the process stops. This is indicated in the model by the activity *ArtificialEndTask*. This is also visible in the OR-split after activity “07” (*Second reminder*), representing a flow to either activity “13” (*Judge standard verification*), or activity “33” (*Collection*) or a payment (*ArtificialEndTask*). After the

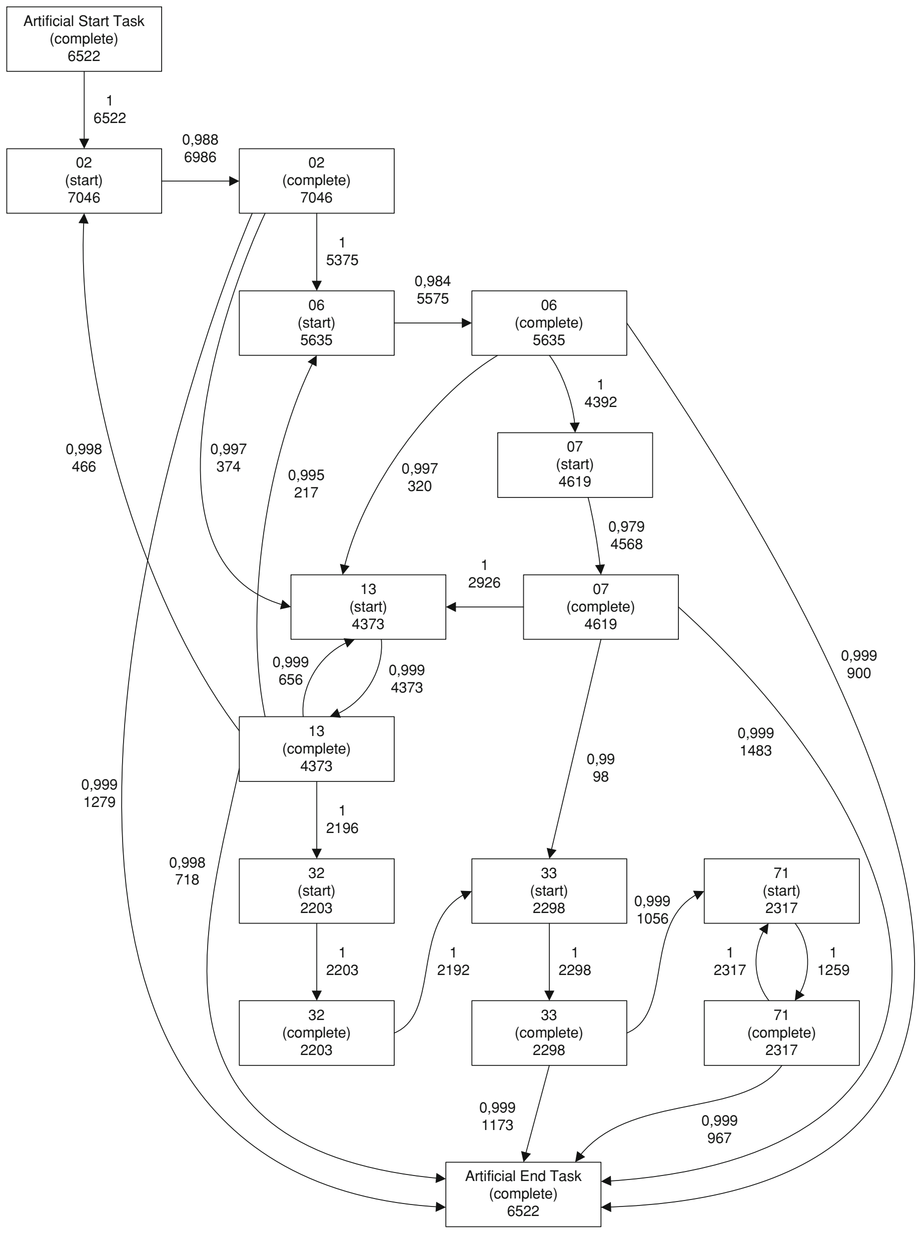


Fig. 8 The process model of collecting fines in case of the Governmental institution, represented as a heuristic net, using parameter tuning

verification activity “13” (*Judge standard verification*), activity “02” (*Initial regulation*) can be reactivated, or a reminder is sent—activity “06” (*First reminder*). Both manual activities “13” *Judge standard verification* and “71” (*Collection to judge*) are in some cases executed multiple times. As they are manual activities, it may take a considerable amount of time to execute them more than once.

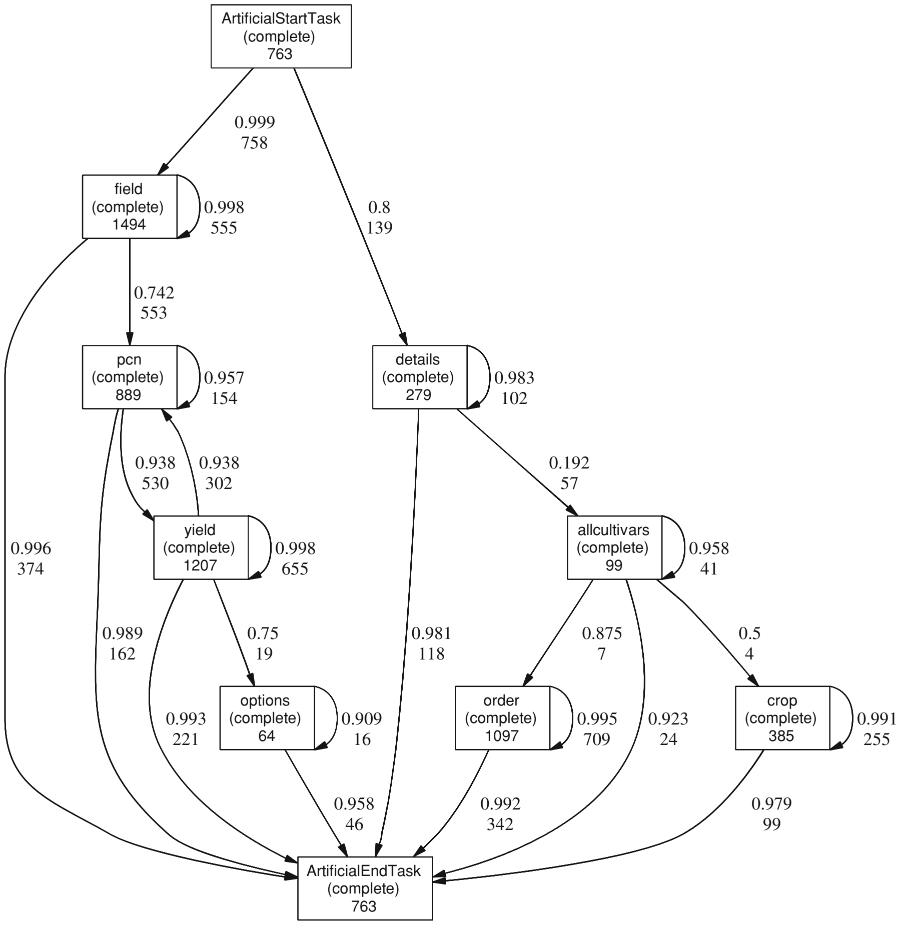


Fig. 9 The mined behaviour corresponding to all agricultural users, using default parameter settings

Based on the model, "the most common pattern" can be obtained, which consists of *ArtificialStartTask* → *Initial regulation* (6,522), *Initial regulation* → *First reminder* (5,573), *First reminder* → *Second reminder* (4,378), *Second reminder* → *Judge standard verification* (2,894), *Judge standard verification* → *Judge: collection* (2,216), *Judge: collection* → *Collection* (2,215) and *Collection* → *ArtificialEndTask* (1,245). A most common pattern as shown here, may reveal interesting information about the process. It can be observed from the model that the majority of the fines require both a first and second reminder before the fine is actually paid, although the reminders serve as an exception in case a fine is not paid within 8 weeks.

3.2.3 The process model in case of DSS

In Fig. 9, the result of applying the ProM heuristic mining algorithm with default parameter setting is shown. The focus is to grasp the patterns of navigation from one page to another (we use the FromPage field of the log file, see Table 1). The file consists of 763 user sessions, belonging to 501 individual users. The rectangles refer to transitions or activities, in our case to page names, such as *Field*, *Order*, etc. The number inside every rectangle shows how many times a page has been invoked. The causality/dependency measure and the frequency of occurrences are also reported besides each arc. For instance, in Fig. 9, *Field* page is immediately followed 553 times by the *Pcn* page, with a dependency measure of 0.742, and 374 times by *ArtificialEndTask*, with a dependency measure of 0.996.

Figure 9 can be interpreted as follows. We can determine "the most common pattern" which consists of *ArtificialStartTask* → *Field* (758), *Field* → *Pcn* (553), *Pcn* → *Yield* (530), and *Yield* → *ArtificialEndTask* (221). The sequence $Field \rightarrow Pcn \rightarrow Yield$ is actually the "prescribed" order of pages in the DSS system, e.g. the order in which pages are presented.

We notice the reversed link $Yield \rightarrow Pcn$, which may suggest that users change PCN values to observe the impact on calculating the yield (the higher the values of PCN, the smaller the yield). Also noticeable is the highly frequent self-recurrent link to $Yield$ (655).

The interpretation of these findings is that agricultural users in general use the prescribed sequence of page invocation. The directed graphs reveal that the invocation of 'Action' pages predominate. In Fig. 9, the sum of incoming arcs from 'Action' pages (Field, Pcn, Order) to activity *ArtificialEndTask* is $374 + 162 + 342 = 878$. This value is much larger than the sum of incoming arcs from 'Target' pages (Yield, Crop) to the activity *ArtificialEndTask*, which is $221 + 99 = 320$.

The first conclusion is that pages from the category 'Target' are visualized significantly less than 'Action' pages. This is a disappointing result, given the fact that the main goal of the DSS is to provide detailed information about yield. Second, after each page type a significant number of sessions stop (in Fig. 9, 374 after page $Field$, 162 after page Pcn). However, we also observe a relevant activity whenever users do not end their sessions prematurely in the earlier stages and proceed to visualise 'Target' pages. They repeatedly recall the *Yield* page, and they revisit pages, which are not directly linked in the prescribed order (the reverse link $Yield \rightarrow Pcn$). This fact illustrates that whenever users reach a 'Target' page, their interest may rise. The number of visits to the *Details* page is surprisingly high: 279 times, and even forms a path containing only one page (e.g. the path *ArtificialStartTask*, *Details* and *ArtificialEndTask* (see Fig. 9). This suggests that users are interested in information about different cultivars.

In the DSS case, the quality of the mined process model is initially represented by a 'Continuous Semantics fitness' measure of 0.28. Attempting to reduce the noise, Fuzzy miner identified only one outlier, which was removed from the log. Increasing the *positive observations* parameter to 200, as in the previous case, results in an increase of the continuous semantics fitness to 0.50. An interesting observation is that the change of the *positive observations* parameter caused a few self-loops to vanish (see Fig. 10). The fact that we could increase the quality of the process model only to 0.50 lead us to conclude that there is no unique generic navigation pattern, but different navigation patterns exhibited by different types of users.

3.3 Analysing performance

For all the three cases, so far we have obtained process models that show a good fit with the log. Therefore, we can rely on them as being good representations of the process flow in reality. The heuristic nets representations of the process models can be converted to Petri nets. Using the Petri net formalism, a performance analysis focusing on throughput times and determining bottlenecks can be performed. In the following, we show the performance analysis only in the Gas company and the Governmental institution (the third case study has a different redesign focus with respect to goal achievement).

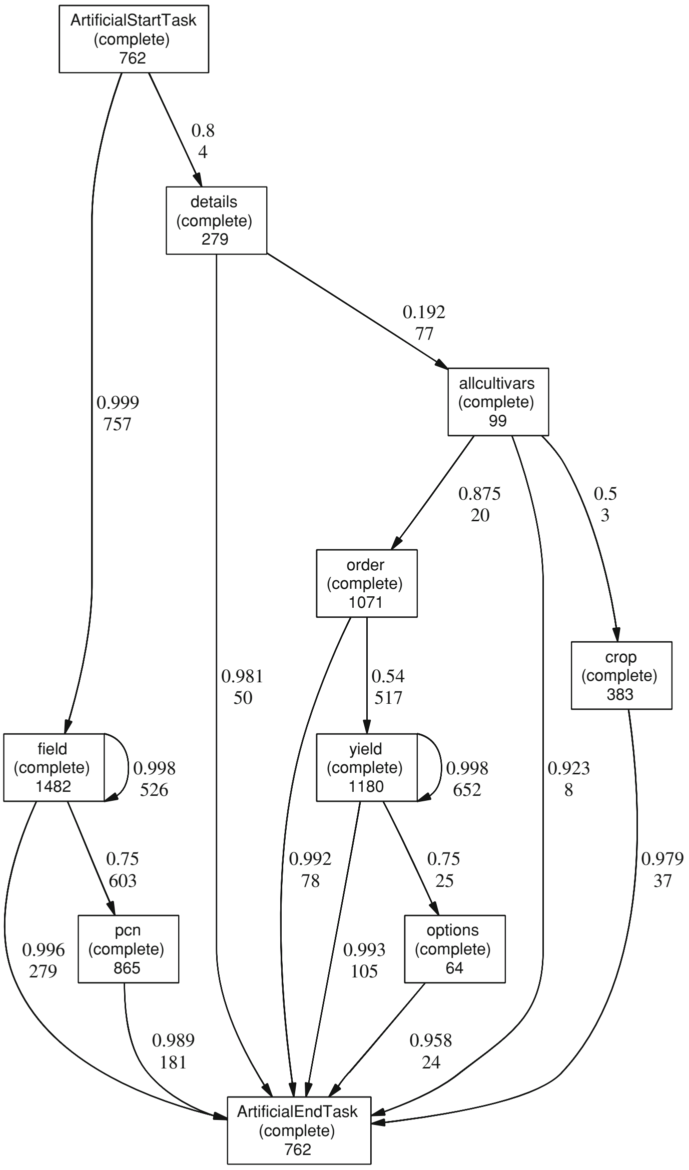
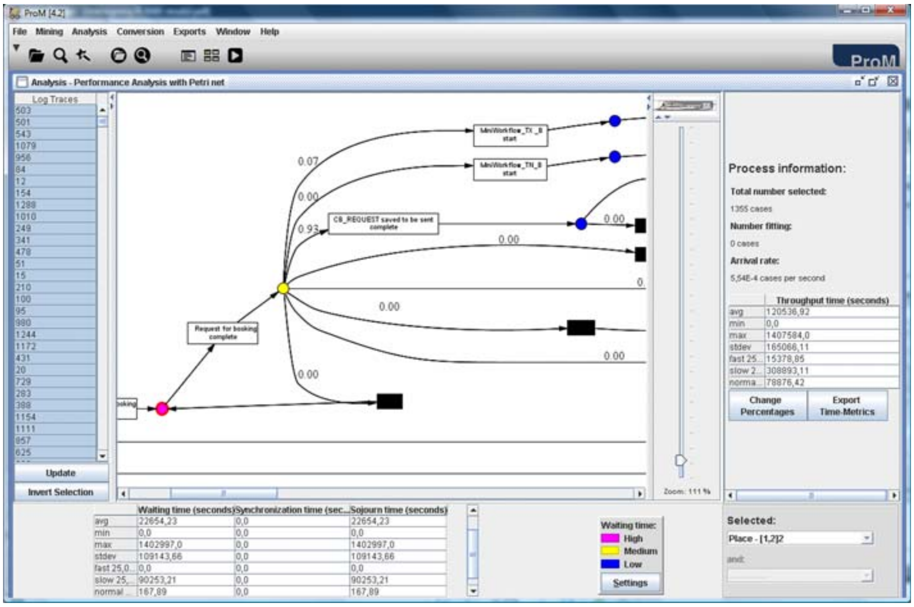


Fig. 10 The mined behaviour corresponding to all agricultural users, using parameter tuning

For the Gas company, an example of bottleneck analysis on activity level is shown in Fig. 11, using the Petri net representation. The bottlenecks in the process are marked with colours to indicate their severity. On the right-hand side, the overall throughput time of the entire process is shown. On the bottom of the screen, information about the throughput time of individual places and transitions are shown, such as waiting time, arrival rate, etc.

Given this information, different alternatives for redesign are possible, such as shortening throughput time or eliminating bottlenecks. Activities that show a large average throughput



Process information:

Total number selected:
1355 cases

Number fitting:
0 cases

Arrival rate:
5,54E-4 cases per second

	Throughput time (seconds)
avg	120536,92
min	0,0
max	1407584,0
stdev	165066,11
fast 25...	15378,85
slow 2...	308893,11
norma...	78876,42

	Waiting time (seconds)	Synchronization time (sec...	Sojourn time (seconds)
avg	22654,23	0,0	22654,23
min	0,0	0,0	0,0
max	1402997,0	0,0	1402997,0
stdev	109143,66	0,0	109143,66
fast 25,0...	0,0	0,0	0,0
slow 25,...	90253,21	0,0	90253,21
normal ...	167,89	0,0	167,89

Fig. 11 Example of assessing waiting time per bottleneck

time with a high standard deviation are potential candidates to be redesigned. Redesigning these activities can either take place by increasing capacity or, in the case of a manually executed activity, performing this activity automatically. By gradually decreasing the level of granularity in the vicinity of a specific bottleneck can help to identify the cause of the outliers and decrease the severity of the bottleneck.

In the Gas company case, for example, some very time consuming process instances are caused by repeatedly occurring error-messages, which result in a loop of the Miniworkflow TX _O high-level activity that was executed repeatedly until success. Some other details can be revealed as well. For instance, in some cases, the Miniworkflow is switching from automatic execution (Miniworkflow TX_O) to manual execution (Miniworkflow TX_B). A similar situation can be identified for the 'Request for Booking' activity, which causes a severe bottleneck. The booking process is not entirely automatically supported for some specific capacity requests. The long throughput time in these cases can be attributed to the occasional manual execution.

Concerning fine collection at the Governmental institution, a similar situation occurs. The manual activity "71" (*Collection to judge*) causes large outliers in the overall throughput time in some cases. Repeatedly invoking activities with long throughput times may have a huge impact on the overall throughput time. Hence, the entire process may significantly benefit by a sole optimisation of these specific activities.

4 Simulation of the As-Is and the To-Be process models

An essential step of the redesigning methodology is the simulation phase of our proposed methodology. This phase consists of two separate steps:

1. Create a (simulated) Petri net with CPN Tools, representing the As-Is process model, in order to check the fit between the simulated process model (M2 in Fig. 1) and the process model based on the raw data (M1 in Fig. 1).
2. Create a (simulated) Petri net with CPN Tools, representing the To-Be process model, in order to check its performance (M3 in Fig. 1).

A Petri net is a graphical and mathematical modelling tool based on graph theory. Concurrent and asynchronous behaviour of discrete systems can be modelled using Petri nets. In addition to the primitives for process interaction provided by Petri nets, CPNs offer the definition of data types and handling of data values [8].

CPN Tools is a powerful tool for editing, simulating, and analysing CPNs, and it can also be used for modelling Petri nets. In order to perform experiments with Petri net models, the ProM framework is equipped with a plug-in, which imports logs based on Petri nets created with CPN Tools. The ProM CPN library is a set of predefined functions that supports the creation of MXML files based on the simulated CPNs. This library was created to facilitate experimenting with various process mining techniques and plug-ins. The details of this plug-in are extensively described in CPN Tools [2].

The necessary steps for performing the simulation are described in the following subsections. In Sect. 4.1, the proposed approach is described.[4] In the next two subsections we illustrate this approach with our case studies. We restricted our discussion of the simulation only to the cases of the Gas company and of the Governmental institution. This selection was done depending on the possibility to simulate performance indicators, namely the throughput time and bottlenecks, such that a comparison between the As-Is and the To-Be situation can be made. However, in case of a DSS, the simulation of goal achievement is not relevant, rather it should be checked via a pilot study whether the To-Be situation shows improvements.

4.1 Create process models using CPN Tools

Modelling a Petri net in CPN Tools requires a specific structure to initialise the Petri net simulation. Consequently, it should be present at the beginning of any model, regardless of the case or process being modelled. In this structure, the case ID's (individual process instances) are generated and put into the system at an inter-arrival time of 100 s. A graphical representation is shown in Fig. 12. Alternatively, it is possible to use a function instead of a constant to insert an arrival rate with a certain distribution. In this example, the process instances are created here at a certain arrival rate, which is fixed. In practice, there are business processes, which can start with different activities or end with different activities. These processes do not have a fixed start and end event. In ProM this is solved by providing each process with constructs such as *ArtificialStartTask* and *ArtificialEndTask*. These constructs will be generated in the simulated model as well. Furthermore, they allow for a simple analysis of the overall throughput time.

The code statements used to construct the CPN network are described below.

– $if\ id > 1{,}000\ then\ 1`(id + 1)\ else\ empty$;. In this sequence the case id's are 'generated'. That is, this place and the transition Generator create tokens with a unique number (id), which represents the process instance number. Generator is putting the token back to the initial place under the following condition: if id $>$ 1,000 then a new token is created with value id + 1. Else no token is returned. Therefore, a maximum of 1,000 case id's are created and put into the system.

[4] The material presented in Sect. 4.1 has been already published in [22]. However, in order to provide a complete methodology, the text is reproduced.

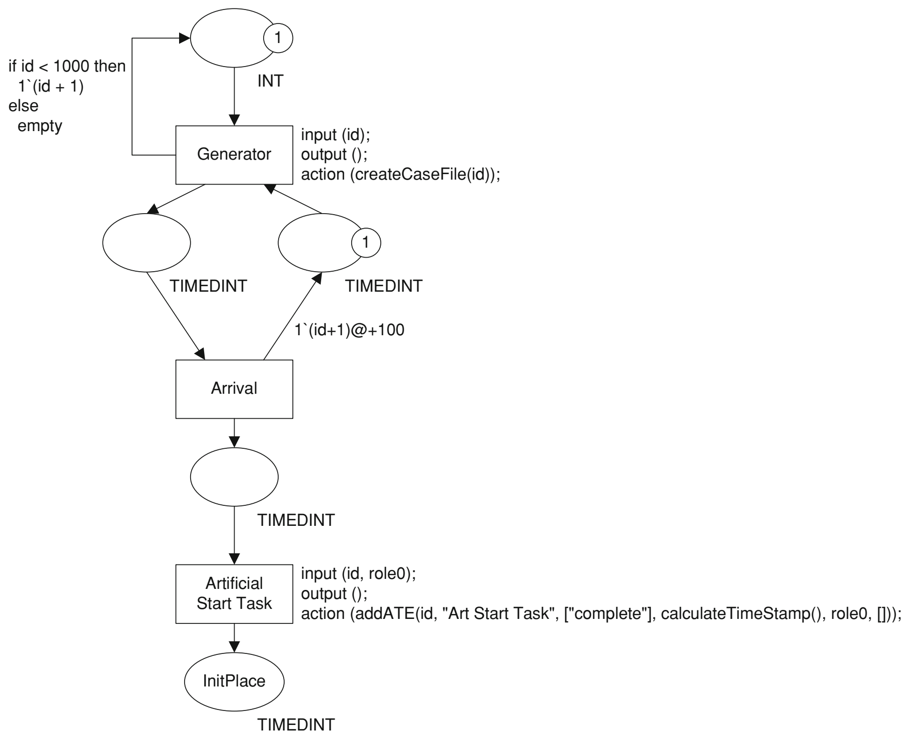


Fig. 12 Example of the first part of Petri net created with CPN Tools

- $input(id)$; $output()$; $action(createCaseFile(id))$; During simulation, a new case-file is created for each new case id. Each id generated by Generator is put into a new XML-file, which later can be converted to a MXML-file using ProM*Import*.
- $1`(id + 1)@ + 100$;. This statement consists of several parts. A' is the anti-place of A. That is, A' serves as a capacity-constraint. A' contains only 1 token. Therefore, A can only contain 1 case id at a time. Transition Arrival is putting a token with id + 1 back to A', thereby releasing a new case-id from Generator. The part @+100 means that every-time a new case-id is released a time is added of 100 to the token. This statement serves, therefore, as an interarrival time.
- $input\ (id, role0)$; $output\ ()$; $action\ (addATE(id,$ "$ArtStartTask$", ["$complete$"], $calculateTimeStamp(),\ role0,\ [])$); A new entry is added to the XML-file, containing id as process instance, "ArtStartTask" as activity, "complete" as eventtype, calculate-TimeStamp() as timestamp, and role0 as originator.

As was mentioned earlier, the analysis at the Gas company began at a high level, before gradually moving to a lower level. In order to allow for such adaptations, the high-level processes should be modelled as hierarchical transitions. In this way, the details can be changed, while preventing the high-level structure from being affected. Hierarchical modelling is not required in cases where the level of granularity is homogeneous among all activities and it is unnecessary to adjust the level of detail.

Figure 13 shows the subnet of high-level activity A. Both the start and complete events of this activity are recorded. The processing time is 100 s, but it is possible to introduce a

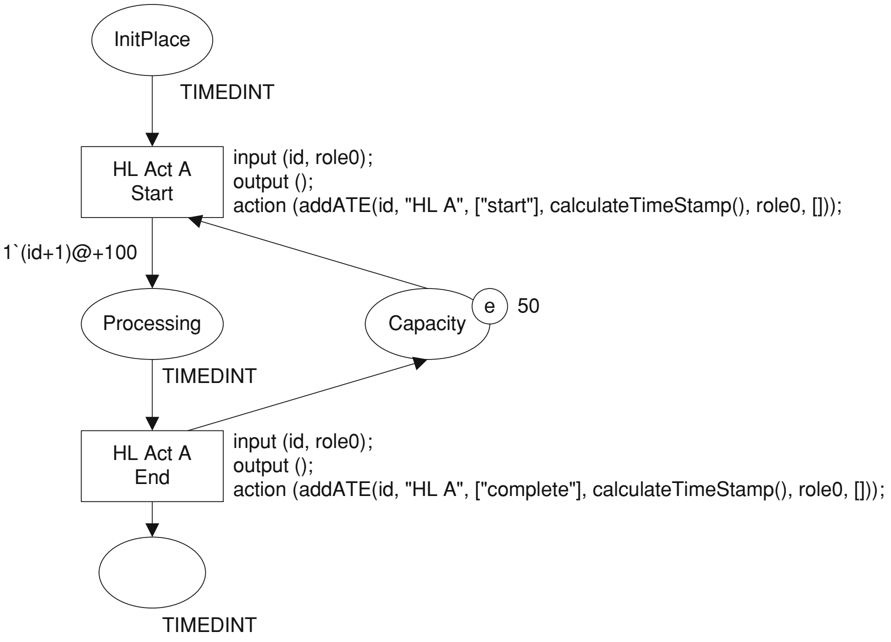


Fig. 13 Example of a high-level activity modelled in CPN Tools

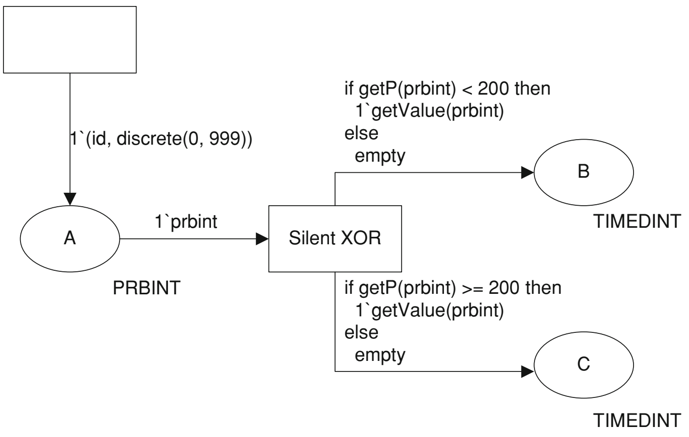


Fig. 14 Example of XOR-split modelled in CPN Tools

function to represent a certain distribution. Finally, in this particular example, the capacity of this activity is set to 50. That is, only 50 process instances can be processed at a time. In the real world, this implies that a certain service can only process 50 requests at a time. This variable depends on the process and can, therefore, be different for each process or activity. The place ‘Capacity’ serves as an anti-place containing 50 uncoloured tokens.

A process may contain many XOR-splits (see Fig. 14). For example, two activity sequences may occur: ABC and ACB. Suppose that the sequence ABC occurs 100 out of 500 times and the sequence ACB occurs 400 out of 500 times. This implies a relative frequency of 20 versus

80%. As standard Petri nets only support OR-splits, in which activity sequences occur on a random basis, a way to model an XOR-split has to be devised. In CPN Tools, a datatype with two tuples, containing a datatype that is required in the token, and a probability integer should be created. Suppose, for instance, that the token contains an integer representing the process instance. Before the XOR-split, this datatype should be converted to a datatype consisting of two tuples: one containing the original value, and one containing the a random value between 0 and 999. Based on this value, a decision can be made to which place a token should be produced.

Figure 14 illustrates the above discussion. Before place A, the token id is converted to a datatype containing an integer and a random value. After transition Silent XOR the token can either go to place B with a probability of 20.0%, or to place C, with a probability of 80.0%.

4.2 Modelling distributions

The average throughput times of the activities along with some parameters can be obtained from ProM. In many cases, the throughput times can be assumed to be exponentially distributed. Since extreme outliers may be present in business processes, a more detailed analysis of the throughput times distributions is recommended, in order to model the process as accurate as possible. Distributions that are not provided as standard function by CPN Tools can be approximated using combinations of other functions. However, CPN Tools does not support variables containing real numbers. For this reason, it is not possible to create a custom function to describe the distribution, which implies that a function has to consist of multiple standard supported functions. A more thorough analysis of the exact distribution might reveal that the distribution for a particular activity consists of several standard distributions, defined for a domain. For instance, 70% of the cases follow a normal distributed with mean *a*, while the remaining 30% are exponentially distributed with mean *b*.

These types of distributions can be modelled in CPN Tools following three steps:

1. Create list of throughput times (using query). For the activity to be approximated, create a list of all the throughput times for that specific activity. Throughput times can be obtained by creating a list with the differences between the start and complete event of the activity.
2. Determine distribution, mean etc. (using graphs). Next, the distribution form can be obtained by plotting the data into a graph. By this way, the family of functions can be determined. Furthermore, the mean, maximum and outliers are to be determined, in order to obtain the required parameters.
3. Create approximation in CPN Tools. The distribution function, along with the parameters, can be inserted into ProM. Due to uncertainties that may occur from the distribution approximations, it is advisable to create a temporary Petri net for testing purposes, containing an activity with the derived distribution. This small temporary Petri net should be executed with ProM and give the same results with respect to the throughput time for that specific activity.

4.3 Assessing the fit of the simulated model with the reality model

Once the approximation of the distributions are as close as possible to the real values, the Petri net of the simulated model can be built. The closer the approximation, the more reliable is the prediction of the To-Be model. Since the intention is to obtain a simulated model similar with the initial model based on real data, the following indicators of similarity between models M1 and M2 are proposed (see Fig. 1):

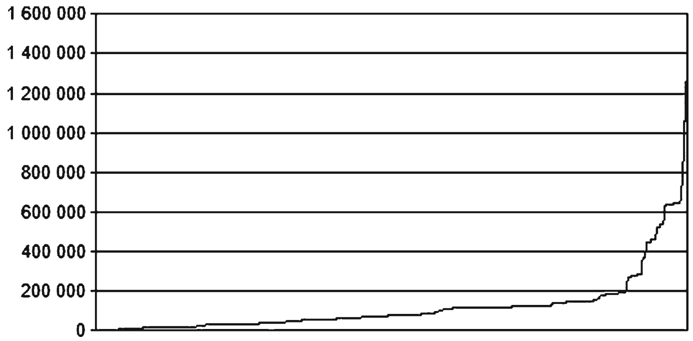


Fig. 15 Gas company throughput time distribution of model M1

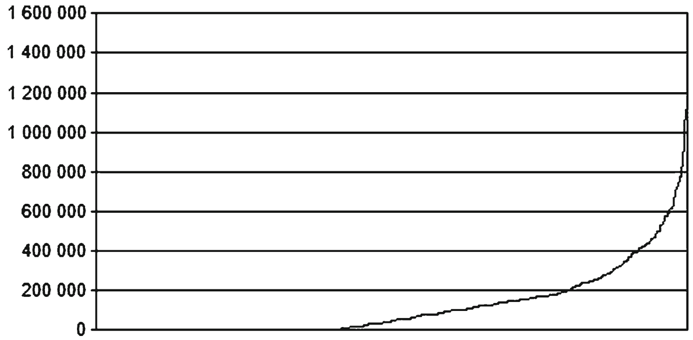


Fig. 16 Gas company throughput time distribution of model M2

- Process flow and semantics. Both models (M1 and M2) should show the same process flow expressed as ordering of activities and Petri nets constructs. Also, with respect to semantics, the activity labels should be identical.
- Throughput times. Throughput times per activity and the overall throughput time of the entire process for the reality process model M1 should be identical with the simulated process model M2.
- Bottlenecks. The location and severity of bottlenecks for process model M1 should be identical with the location and severity of bottlenecks for model M2.

For assessing the fit in case of the Gas company model, the Petri net model created with CPN Tools (M2) is compared with the Petri net model based on real data (M1). First, the process flow and semantics are inspected, and no deviations are found. For example, if a small deviation exists in that respect, it can be solved easily by modifying the XOR-relationships. Second, the throughput time values are inspected, both per activity and per overall throughput time. Deviations between the throughput time of M2 and M1 can originate from multiple causes, such as the complexity of processes or the presence of outliers, which makes it difficult to obtain similar distributions.

Figures 15 and 16 show the difference between the real process and the simulated process, with respect to the distribution of the overall throughput time (in seconds). Third, the number of bottlenecks, their location and severity are inspected, yielding also comparable values.

The following conclusions were drawn after assessing the fit between the simulated model, M2, and the real model, M1, for the case of the Governmental institution. The fines collection

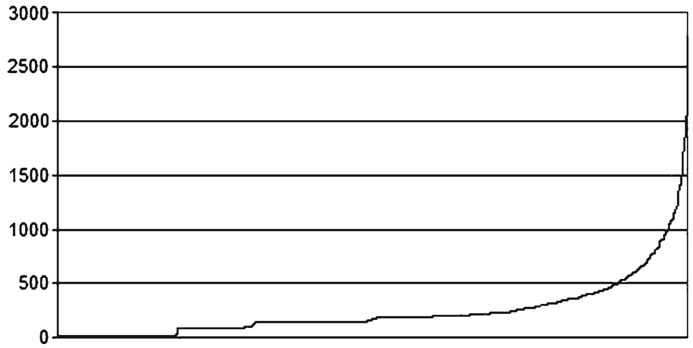


Fig. 17 Governmental institution throughput time distribution for model M1

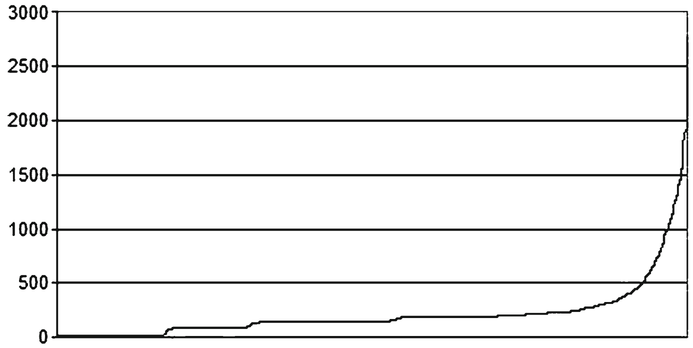


Fig. 18 Governmental institution throughput time distribution for model M2

process does not contain any parallel executions; all process steps are executed sequentially. Furthermore, very few loops exist that cover multiple activities, which implies that the overall throughput time of model M2 is easier to match with the reality model M1 (the existence of loops may determine the appearance of bottlenecks or increase bottleneck severity). Thus, in the Governmental case, the throughput time per activity is primarily influencing bottlenecks. As shown in Figs. 17 and 18, the simulated distribution of the overall throughput time is a close approximation of the reality model M1.

5 Comparing the As-Is process with the To-Be process model

As we have mentioned earlier, the final goal of any redesign project is to improve the current design (As-Is), considering the chosen performance criteria. In this section, we illustrate potential benefits in the To-Be situation that can be acquired by replacing the manually executed activities with the automatic activities, in both the Gas company and the Governmental institution case studies.

In the Gas company case, the ‘Request for Booking’ activity is processed by the front-end application, and usually starts as an automated process. However, in some cases, this activity is manually processed. Manual activities usually have a longer throughput time than the automatic activities, as was shown in the analysis of bottlenecks where manual processes were responsible for extreme throughput times values for ‘Request for booking’.

Table 2 Gas company: comparing 'Request for booking activity' throughput time of As-Is with the To-Be situation

Seconds	RFB As.Is	RFB To.Be
Average	21,099.2	9,102.04
Max	1,203,540.0	394,800.0
Min	0.0	0.0
Standard deviation	81,703.4	37,134.82
Fast 25%	143.36	27.79
Slow 25%	83,373.98	36,053.81

Table 3 Gas company: comparing the As-Is with the To-Be situation, entire process

Seconds	Process As.Is	Process To.Be
Average	109,857.5	83,243.9
Max	1,570,620.0	878,100.0
Min	0.0	0.0
Standard deviation	168,746.15	113,528.23
Fast 25%	463.57	214.72
Slow 25%	330,133.39	243,016.02

As an illustration of a possible redesign strategy, the 'Request for Booking' activity is used to identify the possible performance gains by replacing manually executed activities with automated activities. Two comparisons are performed:

1. Only for 'Request for Booking' activity. The throughput time of the 'Request for Booking' activity where also manual processing occur (the As-Is situation) is compared with the throughput time of the 'Request for Booking' activity, when it is only automatically processed (the To-Be situation), see Table 2.
2. For the entire process. The throughput time for the whole process is calculated, when 'Request for Booking' activity includes also manual processing (the As-Is situation). This value is then compared with the throughput time of the whole process, when 'Request for Booking' activity is only automatically performed (the To-Be situation), see Table 3.

In Tables 2 and 3, different measures of throughput time are shown, namely the average and standard deviation, the maximum and minimum obtained, and the fastest 25% of process instances, respectively the slowest 25% process instances. It is clearly visible that a major improvement can be obtained by replacing manual executions with automatic ones. Table 2 shows that a major improvement has been obtained in the throughput time for the 'Request for Booking' activities, whereas in Table 3 the effects of the throughput time reduction is shown concerning the entire process. Moreover, the amount of outliers is reduced, along with the standard deviation.

For the Governmental institution case, in the heuristics net from Fig. 17, the focus is on the loop surrounding activity "71" (*Collection to Judge*). This iteration points to a manual process, which may consume a considerable amount of time, according to Table 5. Since this activity is related to a severe bottleneck in the process, automating it will result into a decrease in throughput time, as well as a corresponding drop of the throughput time of the entire process. From Table 4, it is visible that a major performance gain can be obtained by automating activity "71". By doing so, the average throughput time of the entire process can

Table 4 Governmental institution: comparing entire process throughput time of the As-Is with the To-Be situation

Days	Process As.Is	Process To.Be
Average	235.41	206.68
Max	2,959.0	2,288.0
Min	7.0	7.0
Standard deviation	275.1	206.25
Fast 25%	26.35	28.5
Slow 25%	585.1	466.24

Table 5 Governmental institution: comparing the As-Is with the To-Be situation: activity 71

Days	71 As.Is	71 To.Be
Average	89.18	–[a]
Max	1,078.0	–
Min	0.0	–
Standard deviation	149.38	–
Fast 25%	0.0	–
Slow 25%	31.81	–

[a] By making activity 71 an automatic activity, its duration measured in days is neglectable, in the To-Be situation

be reduced by nearly a month. Furthermore, the maximum throughput time decreased by more than 22 months.

6 Conclusions

In this paper, we have presented a process mining and simulation-based methodology for redesigning business processes. We proposed a step-by-step methodology for redesign, by processing actual data, what we call a bottom-up approach. Existing generic methodologies are aiming to analyse and design information systems supporting business processes in a merely top-down, prescriptive fashion. For instance, well-known approaches such as Structured Systems Analysis and Design Methodology (SSADM), Soft Systems Methodology (SSM), and GRAI adopt a prescriptive approach [1]. However, for re-engineering purposes, such techniques do not use data resulting from real process executions, which would reflect the actual process flow. Such a comparison between the prescribed and the actual process model would be very useful for providing the basis for process redesign.

The proposed methodology was illustrated with three distinct case studies, each of them exhibiting different characteristics. In Table 6, a summary of issues observed from these three case studies is shown. The first step concerning the finding of relevant 'Performance criteria' leads to same indicators in case of Gas company and Governmental institution, namely throughput time and amount of bottlenecks. These are examples of quantitative performance measures, which are relevant in many situations. Companies are striving to redesign their processes in order to make them as efficient as possible. Concerning the DSS case, goal fulfillment is a qualitative performance criterion, which needs more elaborated work to be operationalised.

The second step, related to 'Mining As-Is process model', consists of several substeps. The substep of 'Logging and data processing' demanded a lot of effort and domain knowledge

Table 6 Summary of methodology steps

Steps	Gas company	Governmental	DSS
1. Performance criteria Section 2	Throughput time, Bottlenecks	Throughput time, Bottlenecks	Goal Fulfillment
2. Mine As-Is process Section 3			
2.1 Logging, data processing Section 3.1	Merging, converting	–	In-build functionality
2.2 Create process model Section 3.2	Heuristic net	Heuristic net	Heuristic net
2.3 Check conformance Section 3.2	CSF = 0.99 Fuzzy miner, HN Tuning	CSF = 0.97 Fuzzy miner HN tuning	CSF = 0.50 HN tuning
2.4 Performance analysis Section 3.3	Throughput time, Bottlenecks	Throughput time, Bottlenecks	–
3. Simulate the As-Is and the To-Be process Section 4			–
3.1 Simulate distributions Section 4.2	Throughput time,	Throughput time,	–
3.2 Replicate bottlenecks Section 4.3	Bottlenecks	Bottlenecks	–
4. Compare As-Is and To-be	Replace manual With automatic	Replace manual With automatic	
Gain Section 5	RfB: Shorter TT WP:shorter TT	“71”: Neglectable TT WP: shorter TT	Determined by user

CSF Continuous Semantic Fitness measure, *HN* Heuristic Net mining algorithm, *RfB* Request for Booking process, *TT* Throughput Time, *WP* Entire Process

in case of the Gas company. In particular, the different levels of granularity, at which data were logged, made merging and converting to a suitable format for mining very challenging (for details, see [22]). Concerning the Governmental institution case, no major difficulties were encountered during the data processing. The DSS had an in-built functionality to log the data in a format that enabled its easy conversion to the format required by the mining tool. Given that data were not available at the needed level of detail, this was a suitable solution for obtaining log data, which might not be always possible. The next substep ‘Create process model’ was performed in the same way for all case studies, by employing the heuristic net modelling algorithm from the ProM framework. The obtained process model is checked for conformance with its log, which is a measure of the quality of the obtained models. A higher conformance indicates a better fit between the model and the data. In all case studies, the metric CSF, with values ranging between 0 and 1 is checked. In case of the Gas company process model, the initial value of 0.33 was subsequently increased to 0.99 by removing outliers that were identified by the Fuzzy miner. The same approach was followed for the Governmental institution, where the initial value was 0.42 was increased to 0.97 by the removal of outliers and tuning the heuristic mining algorithm. Concerning the DSS case, the only improvements were done by tuning the parameters of the heuristic mining algorithm, since no outliers were found. The CSF of 0.50 indicated that the process model did not fit the log to a large extent. The interpretation was that there is no unique generic navigation pattern, but rather different navigation patterns exhibited by different types of users in case of DSS usage. Finally, the measurement of the performance criteria was done for the Gas company and the Governmental institution, concerning throughput time and bottlenecks (in Fig. 1 the P1 block). These measures are then used in further comparisons.

The third step 'Simulation of the As-Is and the To-Be' process models consisted of 'Modelling distributions' for throughput times and 'Replicating bottlenecks'. In both cases, similarity in terms of distributions of throughput time and bottlenecks is obtained.

The final step of the proposed methodology is to compare the performance criteria (throughput time) of the As-Is situation with To-Be situation. In case of the Gas company, a clear gain of automating the manual steps could be obtained—from an average of 109.88–83.24 s for the entire process, and from 21.10 to 9.10 s for the 'Request for Booking' process. For the Governmental institution case, the number of days a case spent in average was 235.41 for the entire process, while in the new To-Be situation this could be reduced to 206.68 days in average. For activity "71", the average number of days is 89.18, and it could be reduced in the To-Be situation to a neglectable execution time (less than 1 day). The gain evaluation of the DSS case is beyond the scope of this study.

We conclude that comparing the performance of the business process in the current situation 'As-Is' and the redesigned business process 'To-Be' led to substantial differences, and it highlights the potential performance gains of the redesign.

The methodology presented in this paper offers an empirically based,"bottom-up" approach, which may complement the top-down prescriptive methods. As a direct consequence, such an approach may be very useful to reduce the risks of complex and expensive Business Process Redesign projects. As further research, we plan to validate the proposed methodology with a more elaborate selection of case studies.

Acknowledgments This work was part of the Casimir project. We would like to thank Peter Baard who facilitated the success of this project, and to anonymous reviewers for their very useful comments and suggestions.

References

1. Aguilar-Saven R (2004) Business process modelling: review and framework. Int J Prod Econ 90(2):129–149
2. CPN Tools (2007) http://wiki.daimi.au.dk/cpntools/cpntools.wiki
3. Davenport T, Short J (1990) The new industrial engineering: information technology and business process redesign. Sloan Manage Rev 31(4):11–27
4. Giaglis GM (2001) A taxonomy of business process modeling and information systems modeling techniques. Int J Flex Manufact Syst 13:209–228
5. Gottschalk F, van der Aalst W, Jansen-Vullers M, Verbeek H (2006) Protos2CPN: using colored Petri Nets for configuring and testing business processes. In: Proceedings of the seventh workshop on the practical use of coloured petri nets and CPN Tools (CPN 2006), vol DAIMI, 579, University of Aarhus, Aarhus, Denmark
6. Hammer M, Champy J (1993) Re-engineering the corporation: a manifesto for business revolution. Harper Collins, New York
7. Hopp W, Spearman M (2001) Factory physics. Irwin, McGraw-Hill, New York

8. Jensen K (1994) An introduction to the theoretical aspects of coloured petri nets. In: A decade of concurrency, reflections and perspectives, REX School/symposium, Springer, London, pp 230–272
9. Johansson HJ (1993) Business process reengineering: breakpoint strategies for market dominance. Wiley, New York
10. Klein M, Methlie L (1995) Knowledge-based decision support systems: with applications in business. Wiley, Chichester
11. Lopez-Arevalo I, Banares-Alcantara R, Aldea A, Rodriguez-Martinez A (2007) A hierarchical approach for the redesign of chemical processes. Knowl Inf Syst 12(2):169–201
12. Măruşter L, Faber NR, Jorna RJ, van Haren R (2008) Analysing agricultural users patterns of behaviour: the case of OPTIRas, a decision support system for starch crop selection. Agric Syst 98(3):159–166
13. Oakland J (1993) Total quality management. The route to improving performance, 2nd edn. Butterworth-Heinemann, Oxford
14. Rozinat A, Mans R, Song M, van der Aalst W (2007) Discovering simulation models. BETA Working Paper Series WP 223, Eindhoven University of Technology
15. Rozinat A, Mans R, Song M, van der Aalst W (2008) Discovering colored petri nets from event logs. Int J Softw Tools Technol Transf 10(1):57–74
16. Ryan J, Heavey C (2006) Process modeling for simulation. Comput Ind 57:437–450
17. Short JE, Venkatraman N (1992) Beyond business process redesign: redefining Baxter's business network. Sloan Manage Rev 34(1):7–20
18. Shyu M, Haruechaiyasak C, Chen S (2006) Mining user access patterns with traversal constraint for predicting Web page requests. Knowl Inf Syst 10(4):515–528
19. Smirnov A, Shilov N, Levashova T, Sheremetov L, Contreras M (2007) Ontology-driven intelligent service for configuration support in networked organizations. Knowl Inf Syst 12(2):229–253
20. Spiliopoulou M, Pohle C (2001) Data mining to measure and improve the success of Web sites. Data Min Knowl Discov 5(1–2):85–114
21. The ProM framework (2007) http://www.processmining.org
22. van Beest N, Maruster L (2007) A process mining approach to redesign business processes—a case study in gas industry. In: Ninth international symposium on symbolic and numeric algorithms for scientific computing (SYNASC 2007), vol 2007. IEEE Computer Society
23. van der Aalst W, van Dongen B, Herbst J, Maruster L, Schimm G, Weijters A (2003) Workflow mining: a survey of issues and approaches. Data Knowl Eng 47(2):237–267
24. van der Aalst W, Weijters A (2004) Process mining: a research agenda. Comput Ind 53(3):231–244
25. van Hee KM, Reijers HA (2000) Using formal analysis techniques in business process redesign. In: Business process management. pp 142–160
26. Weijters A, Aalst W (2003) Rediscovering workflow models from event-based data using Little Thumb. Int Comput Aided Eng 10:151–162
27. Weijters A, van der Aalst W, de Medeiros AA (2006) Process mining with the heuristics miner-algorithm. BETA Working Paper Series WP 166, Eindhoven University of Technology

Author Biographies

Laura Măruşter is assistant professor at the Faculty of Economics and Business, University of Groningen, The Netherlands. She studied Computer Science and received her PhD degree from Eindhoven University of Technology, The Netherlands, with research on a machine learning approach to understand business processes (2003). Currently, her research interests include knowledge management, user modelling and personalisation, induction of machine learning and statistical models, process modelling and process mining. She published various conference and journal papers and book chapters.

Nick R. T. P. van Beest is currently a PhD candidate at the Department of Business and ICT, University of Groningen. He received his Master of Science degree in Economics and Business from the SOM research school, University of Groningen. His main field of research concerns Enterprise Information Systems, model-driven development, service-oriented architecture and ERP and Workflow integration.