VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

Titas SAVICKAS

RESEARCH ON BUSINESS PROCESS PREDICTION AND SIMULATION USING EVENT LOG ANALYSIS METHODS

DOCTORAL DISSERTATION

TECHNOLOGICAL SCIENCES,
INFORMATICS ENGINEERING (07T)

VGTU
LEIDYKLA
Vilnius TECHNIKA 2017

Doctoral dissertation was prepared at Vilnius Gediminas Technical University in 2013–2017.

Supervisor

Prof. Dr Olegas VASILECAS (Vilnius Gediminas Technical University, Informatics Engineering – 07T).

The Dissertation Defense Council of Scientific Field of Informatics Engineering of Vilnius Gediminas Technical University:

Chairman

Prof. Dr Dalius MAŽEIKA (Vilnius Gediminas Technical University, Informatics Engineering – 07T).

Members:

Prof. Dr Habil. Antanas ČENYS (Vilnius Gediminas Technical University, Informatics Engineering – 07T),
Dr Robertas DAMAŠEVIČIUS (Kaunas University of Technology, Informatics Engineering – 07T),
Prof. Dr Habil. Gintautas DZEMYDA (Vilnius University, Informatics Engineering – 07T),
Assoc. Prof. Dr Raimundas MATULEVIČIUS (Tartu University, Estonia, Informatics Engineering – 07T).

The dissertation will be defended at the public meeting of the Dissertation Defense Council of Informatics Engineering in the Senate Hall of Vilnius Gediminas Technical University at **10 a. m. on 18 December 2017**.

Address: Saulėtekio al. 11, LT-10223 Vilnius, Lithuania.
Tel.: +370 5 274 4956; fax +370 5 270 0112; e-mail: doktor@vgtu.lt

A notification on the intent of defending of the dissertation was sent on 17 November 2017.
A copy of the doctoral dissertation is available for review at VGTU repository http://dspace.vgtu.lt and at the Library of Vilnius Gediminas Technical University (Saulėtekio al. 14, LT-10223 Vilnius, Lithuania) and the library of Kaunas University of Technology (K. Donelaičio st. 20, LT-44239 Kaunas, Lithuania)

VGTU leidyklos TECHNIKA 2017-047-M mokslo literatūros knyga

ISBN 978-609-476-076-1

titas.savickas@vgtu.lt

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Titas SAVICKAS

VERSLO PROCESŲ PROGNOZAVIMO IR IMITAVIMO TAIKANT SISTEMINIŲ ĮVYKIŲ ŽURNALŲ ANALIZĖS METODUS TYRIMAS

DAKTARO DISERTACIJA

TECHNOLOGIJOS MOKSLAI,
INFORMATIKOS INŽINERIJA (07T)

VGTU
LEIDYKLA
Vilnius TECHNIKA 2017

Disertacija rengta 2013–2017 metais Vilniaus Gedimino technikos universitete.

Vadovas

prof. dr. Olegas VASILECAS (Vilniaus Gedimino technikos universitetas, informatikos inžinerija – 07T).

Vilniaus Gedimino technikos universiteto Informatikos inžinerijos mokslo krypties disertacijos gynimo taryba:

Pirmininkas

prof. dr. Dalius MAŽEIKA (Vilniaus Gedimino technikos universitetas, informatikos inžinerija – 07T).

Nariai:

prof. habil. dr. Antanas ČENYS (Vilniaus Gedimino technikos universitetas, informatikos inžinerija – 07T),
dr. Robertas DAMAŠEVIČIUS (Kauno technologijos universitetas, informatikos inžinerija – 07T),
prof. habil. dr. Gintautas DZEMYDA (Vilniaus universitetas, informatikos inžinerija – 07T),
doc. dr. Raimundas MATULEVIČIUS (Tartu universitetas, Estija, informatikos inžinerija – 07T).

Disertacija bus ginama viešame Informatikos inžinerijos mokslo krypties disertacijos gynimo tarybos posėdyje **2017 m. gruodžio 18 d. 10 val.** Vilniaus Gedimino technikos universiteto senato posėdžių salėje.

Adresas: Saulėtekio al. 11, LT-10223 Vilnius, Lietuva.
Tel.: (8 5) 274 4956; faksas (8 5) 270 0112; el. paštas doktor@vgtu.lt

Pranešimai apie numatomą ginti disertaciją išsiųsti 2017 m. lapkričio 17 d.

Disertaciją galima peržiūrėti VGTU talpykloje http://dspace.vgtu.lt ir Vilniaus Gedimino technikos universiteto bibliotekoje (Saulėtekio al. 14, LT-10223 Vilnius, Lietuva) ir Kauno technologijos universiteto bibliotekoje (K. Donelaičio g. 20, LT-44239 Kaunas, Lietuva)

Abstract

Business process (BP) analysis is one of the core activities in organisations that lead to improvements and achievement of a competitive edge. BP modelling and simulation are one of the most widely applied methods for analysing and improving BPs. The analysis requires to model BP and to apply analysis techniques to the models to answer queries leading to improvements. The input of the analysis process is BP models. The models can be in the form of BP models using industry-accepted BP modelling languages, mathematical models, simulation models and others. The model creation is the most important part of the BP analysis, and it is both time-consuming and costly activity. Nowadays most of the data generated in the organisations are electronic. Therefore, the re-use of such data can improve the results of the analysis. Thus, the main goal of the thesis is to improve BP analysis and simulation by proposing a method to discover a BP model from an event log and automate simulation model generation.

The dissertation consists of an introduction, three main chapters and general conclusions. The first chapter discusses BP analysis methods. In addition, the process mining research area is presented, the techniques for automated model discovery, model validation and execution prediction are analysed. The second part of the chapter investigates the area of BP simulation.

The second chapter of the dissertation presents a novel method which automatically discovers Bayesian Belief Network from an event log and, furthermore, automatically generates BP simulation model. The discovery of the Bayesian Belief Network consists of three steps: the discovery of a directed acyclic graph, generation of conditional probability tables and their combination. The BP simulation model is generated from the discovered directed acyclic graph and uses the belief network inferences during the simulation to infer the execution of the BP and to generate activity data during the simulation.

The third chapter presents the experimental research of the proposed network and discusses the validity of the research and experiments. The experiments use selected logs that exhibit a wide array of behaviour. The experiments are performed in order to test the discovery of the graphs, the inference of the current process instance execution probability, the prediction of the future execution of the process instances and the correctness of the simulation.

The results of the dissertation were published in 9 scientific publications, 2 of which were in reviewed scientific journals indexed in Clarivate Analytics Science Citation Index.

Reziumė

Verslo procesų (VP) analizės rezultatai leidžia organizacijoms patobulinti verslo procesus bei įgauti konkurencinį pranašumą. VP modeliavimas ir imitavimas yra vieni plačiausiai taikomų VP analizės metodų. Analizės metodų įvestis – modeliai, sukurti taikant plačiai taikomas VP modeliavimo kalbas, matematinius modelius, imitacinius modelius ir kt. Modelių kūrimas yra ne tik svarbiausias, tačiau ir brangiausias bei ilgiausiai trunkantis analizės žingsnis. Šiais laikais didžioji dalis įmonėse naudojamų duomenų yra skaitmeniniai ir jų pakartotinis panaudojimas gali pagreitinti analizę bei pagerinti jos rezultatus. Šios disertacijos tikslas yra patobulinti verslo procesų analizę ir imitaciją pasiūlant metodą, skirtą iš įvykių žurnalo išgauti verslo procesų modelį ir automatizuotai sukurti imitacinį modelį.

Disertaciją sudaro įvadas, trys skyriai bei bendrosios išvados. Pirmajame skyriuje analizuojami VP analizės metodai. Pirmoji skyriaus dalis skirta apžvelgti procesų gavybos sritį ir išanalizuoti metodus, įgalinančius VP analizę, t. y. skirtus automatizuotai išgauti VP modelius ar juos įvertinti bei prognozuoti procesų vykdymą. Antroje skyriaus dalyje pristatomi VP imitacijos srities analizės rezultatai: nustatyti duomenys, reikalingi VP imitacijos atlikimui, bei nustatyti VP imitacijos metodai ir jų taikymo problemos.

Antrajame disertacijos skyriuje pristatomas naujas metodas, kuris automatizuotai iš įvykių žurnalo išgauna Bajeso tikimybinį modelį ir sukuria VP imitacinį modelį. Bajeso tikimybinio modelio išgavimas susideda iš trijų žingsnių: kryptinio beciklio grafo išgavimo, tikimybinių lentelių išgavimo bei išgautų elementų sujungimo. VP imitacinis modelis yra sukuriamas naudojant išgautą beciklį kryptinį grafą. Imitacijos metu naudojamas tikimybinis modelis VP egzemplioriaus vykdymo prognozei bei veiklų duomenų kūrimui imitacijos vykdymo metu.

Trečiajame disertacijos skyriuje pristatomi metodo eksperimentiniai tyrimai bei vertinamas tyrimų rezultatų tikslumas. Eksperimentuose naudojami žurnalai, kuriuose yra tiek sudėtingų, tiek ir paprastų verslo procesų istoriniai vykdymo duomenys. Atlikti eksperimentai vertina grafų išgavimo tikslumą, vykdomų VP egzempliorių tikimybes, VP egzempliorių vykdymo prognozavimo tikslumą bei imitacijos tikslumą.

Disertacijos rezultatai buvo publikuoti 9 mokslinėse publikacijos, iš kurių 2 publikacijos publikuotos žurnaluose, indeksuojamuose Clarivate Analytics Scientific Citation Index duomenų bazėje.

Notations

Symbols

A	–	Activities
A_e	–	Attribute and value set of nodes connected to a node
C	–	Context
D	–	Evidence
E	–	Log Entries (Chapter 1 Only)
F	–	DAG Nodes
H	–	Hypothesis
I	–	Cases (process instances or traces)
L	–	Labels of events (Chapter 1 Only)
L	–	Event Log (Chapter 2 Only)
M	–	Event Attributes
N	–	Attribute Names
$P(H\|D)$	–	Probability for hypothesis *H* based on evidence *D*
$P(H)$	–	*P*robability for a hypothesis
Q	–	Queue
S	–	*D*ynamic business process simulation model
T	–	Tasks (Chapter 1 and Chapter 2)

T – Traces (Chapter 1 Only)
T_L – DAG over an Event Log L
TS – Transition system
V – Value space of Attributes
V_e – attribute and value set of a node
W – Event Log (Chapter 1 Only)
X – Transition relation (Chapter 2 Only)
Y – State space of a Transition System
Y^{end} – Final States of a Transition System
a – Appropriateness Metric (Chapter 1 Only)
c_E – Log Coverage Metric of a Log (Chapter 1 Only)
c_{LE} – Log Coverage Metric of Labels (Chapter 1 Only)
c_T – Log Coverage metric of Tasks (Chapter 1 Only)
f – Token–based fitness Metric (Chapter 1 Only)
hd^k – Function getting first k elements of a partial trace
l_e – Label of event e (Chapter 1 Only)
l_t – Label of task t (Chapter 1 Only)
l^{state} – State of a process instance
l^{event} – State of an event
$name$ – Function mapping event to a name
α – Function mapping activities to events
β – Function mapping event to a case
γ – Function mapping timestamp to an event
∂ – Function to map event pair to an integer number representing a count of times a directly–follows relation is applicable in a log (Chapter 2 Only)
θ – CPTS for a Node
ϑ – function to map event pair to an integer number representing a count of times a follow relation is applicable in a log (Chapter 2 Only)
μ – Attribute and Value function for an event
ρ – function to map event pair to an integer number representing a count of times events occur in the same log (Chapter 2 Only)
σ – function declaring whether an event can be connected in a graph with another event (DAG building section only)
σ – Partial Trace
Ω – function to map event pair to an integer number representing a count of times a cyclic relation is applicable in a log (Chapter 2 Only)
ω – probability function for $P(V_e|A_e)$

Abbreviations

BBN	–	Bayes Belief Network
BBNGs	–	Business Process Belief Network enGine
BP	–	Business Process
BPA	–	Business Process Analysis
BPI	–	Business Process Intelligence
BPI'12	–	Conference on Business Process Intelligence 2012
BPI'13	–	Conference on Business Process Intelligence 2013
BPI'15	–	Conference on Business Process Intelligence 2015
BPM	–	Business Process Management
BPMN	–	Business process modelling and notation
BPR	–	Business Process Reengineering
BPS	–	Business Process Simulation
CPT	–	Conditional Probability Table
DAG	–	Directed Acyclic Graph
DBP	–	Dynamic Business Process
DBPS	–	Dynamic Business Process Simulation
DBPSim	–	Dynamic Business Process Simulator
DES	–	Discrete Event Simulation
EIMSD	–	University Edict Log
GEL	–	Generated Event Log
GMM	–	Generated Mined Model
GUI	–	Graphical User Interface
ILP	–	Integer Linear Programming
IS	–	Information System(s)
NN	–	Neural Network
SEL	–	Source Event Log
SMM	–	Source Mined Model
SVM	–	Support Vector Machine
UI	–	User Interface
UML	–	Unified Modelling Language
VP	–	Business Process
YAWL	–	Yet Another Workflow Language
XES	–	eXtensible Event Stream
XOR	–	Exclusive OR

Definitions

Cost-based fitness – A metric, where fitness is quantified by the failures of the replayability, where each failure (missing event, skipped event, etc.) has a cost, and the fitness is the sum of the costs.

Event Log – A data set with events depicting historical execution of a business process.

Fitness – A metric, which quantifies the ability for the discovered model to reflect the historical business process execution data, e.g. whether the model can replicate all event sequences, whether the data used in the model appears in the event log and other features.

Prediction – An action which, based on the current state of the business process instance, guesses the follow-up events.

Replayability – The ability to replicate the data in the event log on a Petri model using token play technique.

Trace – A grouping of events in the event log for a single business process instance. Use interchangeable with a term Case.

Contents

[1] The annexes are supplied in the enclosed compact disc.

Introduction

Problem Formulation

Business processes (from now on referred to as BP) are one of the main integral parts of any organisation. Therefore the global competitiveness is constantly driving forward their improvement. Constant efficiency improvement of BPs and the (re-)use of knowledge in an organisation are the leading factors which have an impact on organisation success (Thomas & James, 1990; Trkman, 2010). There is a plethora of methods applicable to BP improvement: traditional operations research-based methods, BP modelling and simulation, process mining and big data analytics. However, BP modelling and simulation are one of the most widely applied methods for analysing BPs and allowing their improvement.

Analysis of BPs using standard approaches such as interviews or regulation interpretation is not always effective and often do not reflect the real processes performed in the organisation. It is so because BPs are dynamic and stochastic by nature (Kellner, Madachy, & Raffo, 1999; Van Der Aalst, Nakatumba, Rozinat, & Russell, 2010). Standard static analysis and modelling approaches are not always suitable for BPs when inexplicit knowledge is to be discovered. Therefore, one of the alternatives is a probabilistic analysis of the mentioned cases.

Process mining has come up in recent years as a research area which attempts to improve the reliability and efficiency of BP analysis. The process mining methods and techniques attempt to solve many tasks, such as process model discovery from event logs, conformance checking, model enhancement, predictive analytics and others. They can greatly improve analysis tasks in an organisation by providing automation, but they solve small tasks or are used for different purposes (Augusto *et al*., 2017; Martin, Depaire, & Caris, 2015).

Even if there are many methods for analysing and modelling BPs, there is still a gap in existing state-of-the-art, because: the analytic models which are data-focused ignore BP properties; the static models ignore stochastic properties and are focused on BP static behaviour; the simulation models require much manual labour.

To sum up, the automated discovery of inexplicit and stochastic behaviour of the processes to facilitate analysis is researched, but it remains a problem that has to be solved. The thesis presents a study for the transformation of event logs about BP historical execution into probabilistic BP model to facilitate decision making and simulation model generation, thereby reducing the time it takes from conception to initial BP simulation or decision support.

Importance of the Thesis

The use of BP management methods and tools is widely accepted as mandatory in all organisations to achieve any success in the market. It is expected that the market value of BP management will be 13.52B USD by 2021 (Statista, 2017), which is more than twofold increase from estimated 6.9 billion in 2016 (Statista, 2017).

The reuse of data generated in information systems owned by organisations has become the *de facto* area which creates the biggest value in modern organisations. It has been shown that the use of historical BP execution data to support BP management can be done by employing process mining techniques. The use of process behavioural data can provide value to organisations by facilitating conformance checking, process model discovery or decision support for currently running cases (Griffeth *et al*., 2000).

While many organisations have developed analytic, BI and performance management capabilities to deliver quality information to business users with increasingly sophisticated tools, their ability to align these tools with key BP and to understand how to drive strategic-level business transformation has not matured at the same rate (Oestreich & Chandler, 2015).

There are many techniques and methods to solve individual tasks required for BP analysis or simulation (Augusto *et al*., 2017; De Weerdt, De Backer,

Vanthienen, & Baesens, 2012; Martin *et al*., 2015), but there is limited research dedicated to investigating the path from data in information systems (IS) regarding BP behaviour to BP analysis and simulation.

For the reasons mentioned above this thesis is dedicated to developing a method which could discover a model from event log describing process behaviour to be used in BP analysis, such as for decision support, simulation and others.

The Object of Research

The object of research is the process of predicting and analysing business process behaviour using business process (simulation) models discovered from event logs.

The Aim of the Thesis

The main aim of the thesis is to improve business process analysis and simulation by proposing a method to automatically discover a business process model and generate simulation models from an event log.

The Tasks of the Thesis

In order to achieve the goal, the following tasks must be performed:

1. To review the state-of-the-art in the process mining research area and identify the drawbacks of existing techniques for business process analysis.
2. To review business process simulation techniques and identify existing gaps preventing it for successful use.
3. To propose a method for discovering a business process model from an event log and generating simulation model from discovered business process models.
4. To experimentally validate the proposed methods with synthetic and real data.

Research Methodology

To achieve the goal, the following research methods are employed:

1. The exploratory research method is used while studying the object of the research and reviewing the state-of-the-art.

2. The constructive research method is employed to develop and test the proposed methods for creating probabilistic BP model and its transformation to a simulation model. The prototypes for the proposed methods are implemented using C# programming language, .NET framework and WPF libraries, and experimental evaluation is done using the prototype and PROM tool.

Scientific Novelty of the Thesis

The main scientific contributions of the research are the following:

1. The proposed novel method can discover Bayesian Belief Network from an event log for BP probabilistic analysis. The method relies on a novel algorithm for discovering Directed Acyclic Graph from an event log which eliminates loops from a graph during discovery without losing semantic correctness. The presented method allows decision support by inferring the probability of the currently executed process instances and predicting future events.
2. The proposed novel method can automatically create an initial BP Simulation model. It transforms a Bayes Belief Network, representing a BP, into Dynamic Business Process Simulation model and can simulate BPs with high fitness.

Practical Values of the Research Findings

The proposed novel method can be used for decision support, i.e. the discovered Bayes Belief Network can assist in detecting anomalies and predicting BP execution, including possible data of the currently running process instances.

The proposed method for transforming Bayesian Belief Network into Dynamic Business Process Simulation Model allows business analysts to generate initial BP simulation models which then could be used in BP analysis. This application allows business analysts to save the time required for simulation model creation.

The method was implemented in prototype tools – BBNGs for creating Bayesian Belief Network from an event log and it itself has been integrated into DBPSim BP simulation software. The prototype tools require only a valid XES file, thereby, with small improvements, practitioners could apply it in organisations.

The Defended Statements

The defended statements of this doctoral thesis are as follows:

1. The proposed method discovers Bayesian Belief Network from an event log and eliminates during the discovery of DAG. The method allows inference of probabilities for events with average 63–98% precision and the prediction of events for the current BP instance with good immediate and interval prediction rate ranging 71–87% for the event logs used in the experiments.
2. The proposed method automatically generates Dynamic Business Process Simulation model from the discovered Bayes Belief Network and, when the model is imitated, it can represent the underlying BPs with replayability cost-based fitness ranging 58–98% for generated BP model versus source experiment event logs.

Approval of the Research Findings

The results of the dissertation were published in 9 scientific publications. 2 of them are published in reviewed scientific journals indexed in Clarivate Analytics (also referred to as Thomson Reuters) Science Citation Index, and 7 are published in conference proceedings. The author has also made 4 presentations at international scientific conferences:

- 20th International Conference on Information and Software Technologies (ICIST 2014). October 9–10, 2014, Druskininkai, Lithuania.
- Electrical, Electronic and Information Sciences (eStream). April 21, 2015, Vilnius, Lithuania.
- 23rd International Conference on Information Systems Development (ISD2014 Croatia). September 2–4, 2014, Varaždin, Croatia.
- Data Analysis Methods for Software Systems. December 1–3, 2016, Druskininkai, Lithuania.

Structure of the Dissertation

The dissertation consists of an introduction, three main chapters, general conclusions, references, a list of publications by the author on the topic of the dissertation and a summary in Lithuanian. The total scope of the dissertation is 124 pages, 20 equations, 2 algorithms, 23 figures and 12 tables.

1

Investigation of Methods for Business Process Analysis

BPs are at the core of all organisations, therefore constant BP efficiency improvement and the (re-)use of knowledge in an organisation are few of the main factors leading to the success of organisations (Thomas & James, 1990; Trkman, 2010). The need for the improvement led to the creation of approaches such as business process reengineering (BPR) (Guha, Kettinger, & Teng, 1993), Six Sigma (Narasimhan & White, 2001) and nowadays Process Mining (Vossen, 2012). Process mining facilitates improvements of BPs by reusing data in IS to perform BP analysis. Although there are many ways to analyse and improve BP, there are still gaps and drawbacks in the current state-of-the-art. The purpose of this section is to review the state-of-the-art in existing BP analysis methods and identify the drawbacks and gaps in existing research. The contents of this chapter are based on previously published content by the author in (Savickas & Vasilecas, 2017; Vasilecas, Savickas, & Lebedys, 2014; Vasilecas, Savickas, Normantas, Vysockis, & Kalibatiene, 2015).

BPs are not easy to define and, based on perspective, can have many different properties. The most popular definition of a BP is that a BP is a collection of

activities that takes one or more kinds of input and creates an output that is of value to the customer (Hammer & Champy, 2001). The same BP has a goal and is affected by events occurring in the external world or other processes. Ould adds the property that activities are a coherent set and ordered by a collaborating group (Ould, 2005), although this nowadays is not sufficient because processes are often automated in an IS. Workflow Management Coalition in 1999 introduced the term of procedures and resource roles and defined the process as a set of one or more linked procedures or activities that collectively realise a business objective or a policy goal, normally within the context of an organisational structure defining functional roles and relationships (Workflow Management Coalition, 1999). BPs can be simple and consist of only a few sequential activities, e.g. a cashier work at a store, or the BPs be very complex and span many days and could involve multiple people and systems with hundreds of activities and events.

BP modelling and management is not as easy as might it might look at first sight because the BPs can be complex and involve quite a few hard to manage properties (Kellner *et al*., 1999):

- The processes are stochastic and unpredictable. On the one hand, the processes can be unwieldy inside the organisation due to human resource behaviour (Van Der Aalst *et al*., 2010). On the other hand, the processes can interact with other processes inside or outside the organisation that cannot be controlled (Zhang, 2006).
- The processes change over time (Bose, Van Der Aalst, Zliobaite, & Pechenizkiy, 2014) and new behaviour can be introduced without the appropriate documentation.
- Processes contain complex feedback mechanisms where something at the start of the process impacts later parts of the process, e.g. an error in the creation of a contract might lead to an exception and failure to sign the contract. These feedback mechanisms are often unknown or abstracted away in standard modelling approaches.

Nowadays organisations store most of their data in one or more database management systems or other data storage containers. The purpose of the data ranges from computer systems support to business data management. It originates in ISs and is constantly used during the execution of BPs. The data logged in process-aware IS usually involves all observed behaviour, e.g. taken actions, activities initiated, activities completed and others BP execution-related events (Van Der Aalst, Rosemann, & Dumas, 2007). Although this is the case for process-aware IS, a big part of ISs still store data about process execution in an implicit format in relational databases, files, or other hard-to-interpret formats (Pérez-Castillo, De Guzmán, & Piattini, 2011).

There is a clear need for approaches to analyse BPs and facilitate their improvement since BP management is so important to organisations and the improvement of BP is the driving force in achieving success. Also, the approaches need to take into account the unpredictability of BPs and the historical execution data. Based on this, the following research questions are answered in this chapter:
RQ1: What methods and techniques exist that use historical data to facilitate BP analysis, i.e. automate the creation of models, analyse their behaviour and what are their limitations?
RQ2: What is the state of the BP simulation and what methods exist to automate their creation using historical BP execution data?
RQ3: In what ways do the existing BP analysis methods use historical BP execution data and what is their ability to analyse BPs taking into account their stochastic and dynamic features?

1.1. Business Process Management

BP improvement problem is not a new one, although the direct term had not come to popular use until the 90s, when BPR (Guha *et al.*, 1993) came up. The BPR was not a wholesome approach and dealt with individual processes. In the BPR, the primary way to improvement focused on automating BPs (Hammer, Vom Brocke, & Rosemann, 2010). At that point came the idea of workflow management systems (Georgakopoulos, Hornick, & Sheth, 1995) whose main purpose was to transform existing BPs into automated software procedures. Workflow management systems consist of software components to store and interpret process definitions, create and manage workflow instances as they are executed, and control their interaction with workflow participants and applications (Workflow Management Coalition, 1999).

It soon became apparent that simply automating BPs was not the best approach to improve the performance of the processes and an in-depth analysis is required. That is when Business Process Management (BPM) arose. The purpose of the BPM is to take an all-encompassing control of BPs and extend on what workflow management systems facilitate. The BPM does not only deal with automation and control, but it also controls process design, diagnosis and enactment. The full lifecycle of BPM consists of the following stages:

– Design. At this stage, BPs in an organisation are identified, reviewed, validated and modelled. This point in time allows the identification of the performers of the process, activities that are executed and organisational components involved in the processes. The stage has two primary goals – to find and fix execution problems or to improve the inherent design by replacing or by modifying the process.

- Configuration. This stage deals with the implementation of the process, and during this stage, the process implementation is designed, the process is documented and, finally, implemented. The implementation can be done by training appropriate personnel, implementing a process in workflow management system or implementing it in an IS.
- Enactment is the stage where BPs are performed (enacted). At this stage, appropriate software systems control the processes, make sure that processes do not deviate from the documented model and orchestrate different activity performers to ensure the process achieves the intended goal. This stage creates the basis which is further used for evaluation.
- Evaluation. At this stage, the enacted processes are analysed based on identified performance goals, designed benchmarks or based on customer needs. Business analysts or BPM systems use BP simulation, process mining techniques, interviews with performers and customers and look for ways how a process could be improved. This stage is crucial in the BPM because based on the results, the processes should go back to the Design stage and be appropriately modified to reach the designated performance targets.

Process models, which are developed during design and analysis stages, attempt to represent how processes behave in an organisation. Although the models should in detail describe the behaviour, they are limited in expressiveness. First, the models usually focus on specific aspects of a process or domain. For example, the most popular process modelling language is BPMN (Cherdantseva, Hilton, & Rana, 2012), but the process models represented by the BPMN notation do not have clear data object descriptions or resource definitions. BPMN has only the *data object* and *data storage* elements to represent data in the process, but they cannot model in detail what attributes or parameters the *data objects* actually contain in the real-life processes. The resources in BPMN are usually represented by the *pool* and *swimlane* elements, but they also have limitations as they are implicit – the *pools* can be either a role or a process. Therefore the representation is not formal and leads to vagueness.

Another popular process modelling language is the UML activity diagrams, but they have limited control flow patterns and events. Control flow in activity diagrams are limited to start/end events, fork/join elements and signals which represent events.

The usual modelling languages are focused on eliminating unnecessary details and making the models understandable by business people. While this approach allows easier communication between parties, it also causes the models to be informal. The informality is caused by the fact that elements in these languages can be ambiguous (Korherr, 2006). Formal modelling languages, such as Petri Nets, can describe the behaviour of a process in detail, but the models can

quickly become complex and unwieldy when the modelled processes exhibit complex behaviour.

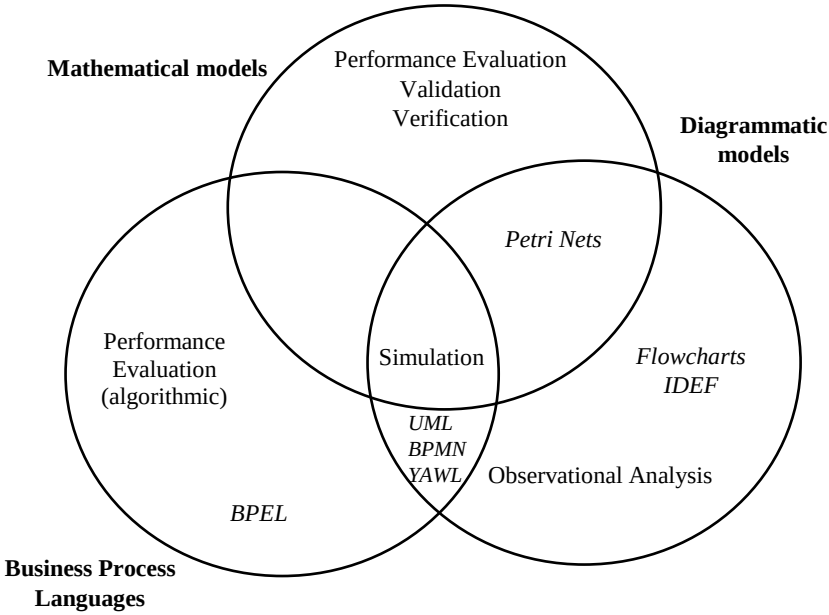


Fig. 1.1. Process modelling types, purpose and examples. Adapted from: (Vergidis, Tiwari, & Majeed, 2008)

Table 1.1. Business process modelling and their execution languages (Korherr, 2006)

BP modelling language	Purpose	Domain	Execution Language
Activity Diagrams	Description, Enactment	Software Engineering	BPEL4WS
BPMN	Description, Enactment	Process Engineering	BPEL4WS
EPC	Description, Analysis	Process Engineering	EPML
IDEF3	Description	Software Engineering	None
Petri Nets	Enactment	System Engineering	PNML

The other problem with BP modelling is that the models can be used for different purposes (Aguilar-Savén, 2004). When processes are modelled for analysis, they can be informal and with the purpose of communication between multiple parties as is the case of BPMN (Cherdantseva *et al*., 2012). When

processes are modelled for execution, the standard process models are transformed to execution languages. Table 1.1 presents the most popular BP modelling languages and their corresponding execution languages.

Finally, when processes are analysed for a specific task, specialised models can be used. If business analysts want to support decision making, they could use probabilistic models or predictive models; when they want to analyse the impact of designed changes, they could use statistical models; when they want to assess an impact of a process change, they could create BP simulation models.

BPM life cycle describes how processes should be dealt with in organisations. To make sure that the implementation of BPs or their changes leads to advancement towards organisation goals, they have to be continually monitored and analysed. Business process analysis (BPA) is the research area on techniques and methods used for validation (Weber, Hoffmann, & Mendling, 2008) and verification (Wynn, Verbeek, Van Der Aalst, Ter Hofstede, & Edmond, 2009) of BPs, and gaining insights or measuring BP metrics (Eicher & Ruder, 2007). For each of the analysis tasks, different approaches can be used, and there are many methods, techniques and tools to perform the analysis.

Historically BPA has fallen under operations management. Operations research is the field of mathematical modelling with mathematical optimisation and queue theory method applications. If these tools were insufficient, simulation methods were used together with discrete event simulation to imitate BP behaviour and analyse it.

Although the mathematical models are sufficient for general analysis tasks, there is a need for specialised methods and techniques in BPA. For this reason, research areas such as process mining, BP simulation, process validation and verification arose.

1.2. Process Mining Techniques for Business Process Analysis

Organizations store most of their data in one or more database management systems or other data storage locations. The data can be related to domain-specific activities or generated during software system activities. In any case, its origin is in IS, and it is generated or used during BP execution. Nowadays, many ISs are process-aware, i.e. they support work-flows and generate a lot of data on how the processes are executed in the form of events (Kalenkova, Van Der Aalst, Lomazova, & Rubin, 2017). Capability to analyse and make useful insights using this data is of paramount importance to business owners because it facilitates the improvement of BPs (Van Der Aalst & Weijters, 2004).

Analysis of BPs executed in an organisation can be performed using Process Mining methods. The purpose of process mining is to use the data existing in IS, extract BP models and facilitate conformance checking, semantic analysis or make useful insights into the performance of the BPs.

Process mining is a research discipline situated between data mining, process modelling and analysis. The idea of process mining is to discover, monitor and improve real processes by extracting knowledge from event logs in IS (Vossen, 2012). It is done by applying different techniques and methods which are classified as follows:

- Discovery. Data in the event logs show the event sequences which represent the execution of the BP. BP model discovery techniques, when applied on this sequence of events, can create a BP model which represents the behaviour of the executed BP instances. The discovered models can be in various notations, e.g. (Coloured) Petri Nets, YAWL, Causal Nets, BPMN and others (Augusto et al., 2017).
- Conformance. When processes are executed in an IS, they are the ones that are performed in an organisation. Organisations usually have documented BPs. Conformance checking techniques attempt to align the contents of the event logs and discovered process models with documented BPs to prove compliance with regulations, standards and contracts.
- Extension (enhancement). Processes documented in an organisation do not always reflect what is happening or can be incomplete. If that is the case, the data on how the processes behave in real-life can be the basis for improving the existing models with additional details. For example, data in the event logs might show which organisational units are performing an activity, and the corresponding documented models could be extended with such information.

1.2.1. Process Model Discovery from Event Logs

Processes, when executed in an IS, leave a trail of their execution. This trail is a set of records for events that occur during the execution of a BP. When the events are connected to a specific BP instance, they form a trace, i.e. a set of events describing a single instance of a BP. These events, once combined in a temporally ordered sequence, describe the sequence and control flow of the process, although this description is implicit because one case cannot describe the sequence of events which applies to all cases.

Process mining techniques, with the purpose to discover BP models, try to process the event sequences to identify the implicit control in the events logs and present the general control flow in the form of BP models (Fig. 1.2). In the example provided in Table 1.2, there is an event log of an insurance claim. The

event log contains data about different BP instances (identified by *Trace ID),* sometimes called traces or cases. Each trace contains a set of events which have occurred during the execution of the BP. The events usually depict an occurrence of the activities, for example, the events *Incoming_claim*, *Register_Claim*, *Initiate_Payment* are activities that have occurred in the process. Each event always has a name, identifying the event, and a timestamp of the occurrence. These data attributes form the smallest set of data that can be called an event log. However, this data is rather limited because it only allows seeing the sequence of activities that have occurred in the BP.

The event logs might also contain domain-specific data, such as, in the case of the example, an organisational resource to identify the agent who has performed the activity and activity-specific data, such as the location of the activity, status of the insurance claim, payment size or others.

Table 1.2. Exemplary event log of an insurance claim process

Trace ID	Event	Timestamp	Organizational Resource	Data
1	*Incoming_claim*	2014.01.05 8:05	*{actor A}*	*{claimant}*
1	*Register_claim*	2014.01.05 8:30	*{actor A}*	*{claim size}*
1	*End*	2014.01.05 13:57	*{actor A}*	*{rejected}*
2	*Incoming_claim*	2014.01.07 13:07	*{actor B}*	*{claimant}*
2	*Register_claim*	2014.01.07 13:13	*{actor B}*	*{claim size}*
		…		
2	*Initiate_payment*	2014.01.10 11:15	*{actor B}*	*{payment size}*
2	*End*	2014.01.10 11:17	*{actor B}*	*{complete}*

There are various techniques for process model discovery from an event log, and they all have their specialised purposes. For example, some techniques can discover process models using specific modelling language such as Petri Nets (Weijters & Ribeiro, 2011) or declarative process models (Maggi, Dumas, García-Bañuelos, & Montali, 2013a). Others are developed to solve a specific task, e.g. guarantee to not overfit the event log (Leemans, Fahland, & Van Der Aalst, 2013).

Discovered process models reflect the BP behaviour exhibited in the event log. Generally, BP models exhibit complex control flow, such as: decisions, *n:m* splits and joins, event-based splits and joins, rule-based split and joins and others (Russell, Ter Hofstede, Van Der Aalst, & Mulyar, 2006). It has to be noted, though, that the event logs only contain event sequences, such as $\{abcd, acbd, abd\}$ which are interpreted in a process model.

Of course, this is a simplification, because the events can contain additional attributes, describing resources associated with events, timestamps, lifecycle types (*start*, *end*, *complete*, *cancel* and others) and other, domain-specific, attributes (Reddy *et al*., 2011; Van Dongen & Van Der Aalst, 2005). It is clear that event logs contain only implicit control flow and it is the task of process discovery techniques to interpret the sequences into a complex control flow. For example, having an event log with traces and event sequences $\{abcd, acbd, abd\}$, it is clear that: a is always followed by either b or c; d always follows b or c; b and c are independent; b can repeat multiple times. A Petri Net best describing the event log is depicted in Fig. 1.3.

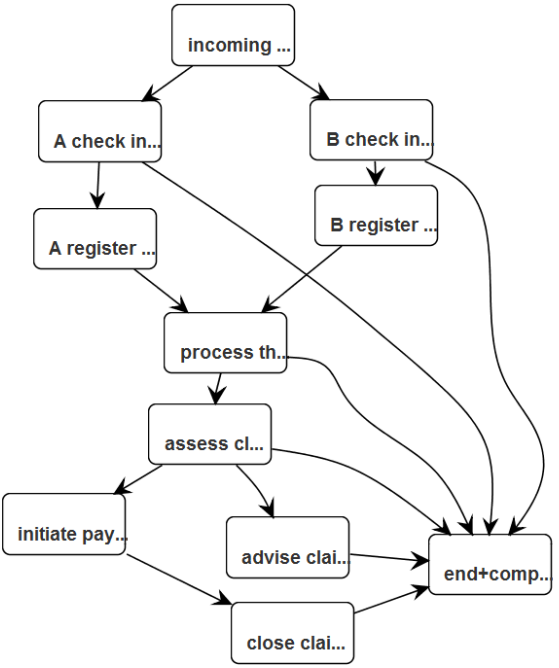


Fig. 1.2. Process model discovered from an event log

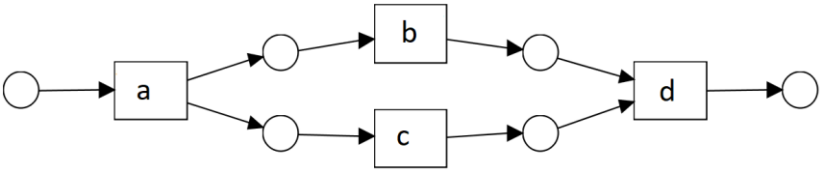


Fig. 1.3. Petri Net discovered from event log $\{abcd, acbd, abd\}$

Also, sometimes the event log contains behaviour that cannot be interpreted in a simple manner and it is apparent that an event must have happened during a process, but it was not observed. For example, when a telephone call is made, some activities might be skipped. This behaviour is explained by invisible events, where the transition between events is apparent, but no observed event is visible to explain it. In that case, usually, a black box in a Petri net is displayed with no label to represent such invisible events.

1.2.1.1. Petri-Net Based Model Discovery Techniques

One of the first and most important process model discovery techniques is α algorithm by Van Der Aalst and Maruster (Van Der Aalst, Weijters, & Maruster, 2004). This technique was the first to discover structured workflow nets. Structured workflow nets are Petri Nets which always have a starting point (a node with no input), ending point (a node with no output) and all places in it are connected. This technique takes an event log, which is assumed to be noise-free, and iterates through all traces in the log. It transforms each event in the log into a place and for each pair of event sequences it adds a transition between them in the discovered structured workflow net. The main fault of the algorithm is that it cannot discover short loops and non-free choice constructs. It also cannot distinguish between multiple events with the same label. Although it has limitations, an improvement was presented by Rebuge and Ferreira in (Rebuge & Ferreira, 2012) and it is was one of the most widely used algorithms.

Another algorithmic approach is Heuristic Miner (Weijters, Van Der Aalst, & De Medeiros, 2006). It applies well-known heuristics and is not noise-sensitive. The method uses relations between events to construct a dependency matrix which is then used to construct a dependency graph. The heuristics miner formally defined well-known relations. Having an event log W with traces T and event pairs $a, b \in T$, the relations can be:

- *Directly-follows*. $a >_W b$ iff $\exists \sigma \in W: \sigma = t1t2 \dots t_n, where\ i \in \{1, \dots, n-1\} \wedge t_i = a \wedge t_{i_{+1}} = b$, i.e. event b at least once in the log was found just after event a.
- *Must-follow*. $a \rightarrow b$ iff $a >_W b \wedge b \not>_W a$, i.e. event a always precedes event b.
- *Independent*. $a \parallel_W b$ iff $\exists \sigma \in W: \sigma = t1t2 \dots t_n, where\ i \in \{1, \dots, n-1\} \wedge t_i = a \wedge t_{i_{+1}} = b \wedge t_{i_{+2}} = a$.
- *Eventually-follows*. $a >>> b$ iff $\exists \sigma \in W: \sigma = t1t2 \dots t_n, where\ i, \in \{1, \dots, n-1\} \wedge i < j \wedge t_i = a \wedge t_j = b$, i.e. event b follows event a in the same trace somewhere in the log, but there could be other events intruding between events a and b.

By using these relations, the Heuristic Miner discovers dependency matrix, where each cell reflects *connectedness* between events a and b and the connectedness is calculated as a metric $a \Rightarrow_W b = (\frac{|a>_W b| - |b>_W a||}{|a>_W b| + |b>_W a| + 1})$. The metric identifies the relation between two events, i.e. if the connectedness is higher, it means that a should be followed by b, while the negative value implies the other way around. $a \Rightarrow_W b = 0$ indicates that events are not connected at all and should not have a direct relation.

Having the dependency matrix, Heuristic Miner builds dependency graph by iteratively looking for the highest connectedness values between events and builds relations between them. The Heuristic Miner takes into account loops and AND-splits and AND-joins while building relations. When representing the dependency graph, it is transformed into Causal Matrix, which can then be transformed into a Petri Net using other known techniques.

The methods mentioned above are built using heuristics. Therefore they rely on expert knowledge. There are also other types of discovery techniques that apply other heuristics-based methods. One of the best-known examples of this is the Genetic Miner (De Medeiros, 2006), which applies the idea of the genetic optimisation algorithms. While this approach is not directly dependable on heuristics, it is much slower and is not guaranteed to succeed or reach optimal scenario. ILP miner (Van Der Werf, Van Dongen, Hurkens, & Serebrenik, 2009) applies integer linear programming methods by representing an event log as a prefix-closed language and applying the language-based theory of regions to discover processes.

To solve the problem that existing methods are not guaranteed to find, or to discover *fitting* or *sound* models, Inductive Miner (Leemans *et al*., 2013) was created. It applies divide-and-conquer strategy to construct a block-structured model from small subsets of event sequences where each block is independent, thereby guaranteeing fitness of each block for the respective event sequences. The Inductive Miner is sensitive to infrequent behaviour because it becomes hard to separate individual blocks, as in the case of infrequent behaviour it is not clear which event sequences are dependent. This problem was solved in a follow-up research that improved the method (Leemans, Fahland, & Van Der Aalst, 2014), but the provided method is still sensitive to infrequent behaviour with regards to XOR-based events.

1.2.1.2. Non-Petri Net Based Process Model Discovery Techniques

Petri Nets, while they are useful due to the availability of formal analysis techniques, are not so useful for visualisation and even more so when the processes are complex. BPMN is the *de facto* BP modelling language that is most widely used, and it is more understandable by business analysts when compared

to other notations. It is for this reason that some approaches choose to discover not Petri Nets, but BPMN models. BPMN notation is difficult though, as it has many elements and can describe complex behaviour (Cherdantseva *et al*., 2012) which is implicit in the event log. Therefore, most of the methods with the purpose of discovering BPMN models use patterns, as is the case for Constructs Competition Miner (Redlich, Molka, Gilani, Blair, & Rashid, 2014). It uses a top-down approach and identifies different event occurrence patterns to discover *Sequence, Loop, Parallelism* and few other predefined common BP constructs. BPMN miner is another approach for discovering BPMN models (Conforti, Dumas, García-Bañuelos, & La Rosa, 2016). It discovers hierarchical BPMN models containing interrupting and non-interrupting boundary events and activity markers by employing functional and inclusion dependency discovery techniques.

BPMN and Petri Nets notations are limited because they require all places and transitions to be connected. Declarative Process notations present processes in a flexible manner, i.e. they only specify either allowed behaviour or not allowed behaviour leaving the exact flow of the process up to the executors (Van Der Aalst, Pesic, & Schonenberg, 2009) (Goedertier, Vanthienen, & Caron, 2015; Van Der Aalst *et al*., 2009). The process models differ from standard models because the start nodes in the model do not necessarily have to be connected with end nodes and the focus of such model is only to declare rules for the flow (Fahland *et al*., 2010). It is for the same reason that process mining techniques for discovery of such models also differ, i.e. they focus on detecting specific patterns to indicate when some activities can be executed. For example, Declare process modelling language (Pesic, Schonenberg, & Van Der Aalst, 2007) provides a template language for temporal logic where the logic rules declare which BP elements can be executed on what conditions.

Apriori algorithm was used by Maggi et al. to identify sets of activities that must follow each other (Maggi, Bose, & Van Der Aalst, 2012). They have also presented a technique for discovering condition-based declare rules in the form $(acitivity_i, activity_j, conditionalRule)$, where the rule indicates data parameters for firing such sequence, e.g. $conditionalRule = \{x \equiv 5; y \neq' John'\}$ (Maggi *et al*., 2013a) and presented a technique for discovering branched constraints (Di Ciccio, Maggi, & Mendling, 2016), i.e. to detect a pattern of a rule, when an activity depends on the execution of two or more other activities.

1.2.1.3. Clustering Based Process Model Discovery Techniques

It is clear that not all events are important in an event log or required to be represented in the discovered model. Clustering-based process model discovery techniques were created to filter out unwanted or "noisy" behaviour during BP

model discovery. An example of such techniques is the Fuzzy Miner (Günther & Van Der Aalst, 2007). Its purpose is to discover simplified models, and fuzzy logic techniques are suited for this task. The main idea of the Fuzzy Miner is as follows:

- Highly significant behaviour should be present in the model.
- Less significant but highly correlated behaviour should be aggregated and hidden in clusters to reduce the size of the model.
- Insignificant behaviour should not be visible in the model for simplification.

The method identifies the significance of single events (unary significance), between event pairs (binary significance) and the correlation between event pair precedence relations.

Another approach (Van Dongen & Adriansyah, 2010) builds on the idea of the Fuzzy miner – it also creates multiple abstraction levels by clustering infrequent events using Simple Precedence Diagrams where nodes in them can have many to many relations with activities in the log.

The approaches mentioned above focus on discovering models that are based on Petri Nets. These methods facilitate formal analysis of the behaviour of the process, but they not so useful for visual analysis.

1.2.2. Process Model Conformance Checking

Discovered process models, or process models in general, should be proven to be valid and must not only exhibit the same control flow between single events (be fitting), but it should also try to exhibit no more of behaviour than is available in the event log. For evaluating the discovered model against the event log, conformance checking techniques are available.

The static metrics calculate the element counts and names in the log and afterwards model and compare them (Rozinat, Van Der Aalst, & Weijters, 2010). For example, the log coverage metric compares how many of the elements in the discovered process model are also available in the event log. The inversed metric is called model coverage, and it compares the count of elements in the discovered model versus the elements in the event log. The metrics are provided in Definition 1.

Definition 1. *Given a set of log entries E, a set of tasks T and a set of labels L, let $l_e \in E \to L, l_t \in T \to\to L, T_v = dom(l_T), L_T = \{l_T(t) | t \in T_v\}$ and $L_E = \{l_E(e) | e \in E\}$, then:*

- *the log coverage metrics are $c_E = \frac{|\{e \in E | l_E(e) \in L_T\}|}{|E|}$ and $c_{LE} = \frac{|L_E \cap L_T|}{|L_E|}$;*
- *the model coverage metric is $c_T = \frac{|\{t \in T_V | l_T(t) \in L_E\}|}{|T_V|}$.*

Other metrics reflect not only conformance between event log and the discovered model using static properties, but also take into account the exposed behaviour. This behaviour is usually benchmarked by using Petri Net Token play approach, which replays the events in the event log against the discovered model (Adriansyah et al., 2011a) or imitates token play in the discovered model to see how well the execution is reflected by the event log.

Fitness metrics describes how much of the behaviour in the log the process model is able to represent. When the model allows more behaviour than the event log exhibits, it is called underfitting and, in contrast, when the model allows not all of the behaviour that is available in the log, it is called overfitting.

Heuristic Miner (Weijters & Ribeiro, 2011) applies the simplest form of fitness calculation – whenever it cannot place some token in the model that is observed in the event log, it treats the model as unfit for the trace. By replaying the full set of traces available in the event log against the process model, it calculates the ratio of the traces that were played unsuccessfully against the total number of traces.

The heuristic miner approach is clear-cut, meaning that one missed transition which appears multiple times in the model can lead to very low fitness. It does not take into account any details of the failure to replay the trace. A more complex model for conformance checking metrics can take into account how complex a discovered BP model is versus how complex the source event log is. Appropriateness is the metric that describes how complex the discovered model is versus how complex the behaviour is, e.g. if it uses many times more elements in the model than there are events or possible paths between events, its appropriateness is low (Rozinat et al., 2010):

Definition 2. *Let k be the number of different traces from the aggregated log. For each log trace i* ($1 \leq i \leq k$)*,* n_i *is the number of process instances combined into the current trace,* m_i *the number of missing tokens,* r_i *the number of remaining tokens,* c_i *the number of consumed tokens, and* p_i *the number of produced tokens during log replay of the current trace. The token-based fitness metric is* $f = \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{k} n_i m_i}{\sum_{i=1}^{k} n_i c_i}\right) + \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{k} n_i r_i}{\sum_{i=1}^{k} n_i p_i}\right)$.

Definition 3. *Let k be the number of different traces from the aggregated log. For each log trace i* ($1 \leq i \leq k$)*,* n_i *is the number of process instances combined into the current trace, and* x_i*the mean number of the enabled transitions during log replay of the current trace (note that invisible tasks may enable succeeding labelled tasks but they are not counted themselves). Furthermore,* T_V *is the set of visible tasks in the Petri net model. The behavioural appropriateness metric is* $a_B = \frac{\sum_{i=1}^{k} n_i(|T_V| - x_i)}{(|T_V|-1)\cdot\sum_{i=1}^{k} n_i}$.

The conformance of the process model against a log cannot always be confidently calculated using the above fitness metrics because not all changes are equal, i.e. missing an activity in the model could be worse than misinterpreting control flow between XOR and OR splits. Also, the fitness metric, as defined above, does not take into account the location where the token was lost – only the number of the tokens, therefore a single missing control-flow link might be reflected as low fitness whereas it is only a single minor error.

To solve this issue, cost-based fitness metrics (Adriansyah, Van Dongen, & Van Der Aalst, 2011b) that assign costs to skipped activities and inserted activities can be used. The skipped activities are activities which are sometimes skipped when the model, requires them to be executed. Inserted activities are the ones which are observed in the log but are not permitted in the model. The costs of the *skip* and *insert* operations depend on the distance from the current location in the replay. This approach replays event log on the model and, once it completes or fails, it finds the nearest complete trace in the log. Once the nearest complete trace is found, the algorithm computes the distance and provides fitness for a single case. For example, if we have a log with a single trace $\sigma = abkc$ and a process model that allows traces $\{abc, abyc\}$, then the closest match between the log and the model is the allowed trace $\{abc\}$. This leads to a single insert operation of the event y, therefore the fitness is the cost (an arbitrary number assigned for this specific case) of the insert operation.

1.2.3. Business Process Execution Analysis

BP event logs can contain much more information than the sequences of events. Each event has additional data attributes that define the context of the event. This data can describe organisational resource related to the event, the timestamp of the event (Reddy *et al*., 2011), the location of occurrence and any other arbitrary domain-specific data, such as product title and weigh, university faculty, student's study year, study programme and other data. All this data could be used for much more than an extension of BPs or evaluating their conformance.

It is for the abundance of such data that process mining techniques extended beyond analysis of process models. Process mining research area has extended beyond the initial model discovery and conformance techniques into BP flow analysis or behaviour parameter detection, e.g. anomaly detection, leftover instance duration prediction, QoS parameter analysis, resource assignment to activities, organisational relations and others. These techniques fall under the umbrella of Operational Process Mining (Senderovich, Weidlich, Gal, & Mandelbaum, 2014c).

1.2.3.1. Business Process Flow Prediction

Process mining has quite a few applications in the analysis of BP behaviour to improve decision making. It has been used for time prediction – Van Dongen et al. use regression equations based on event logs to prepare model for predicting when a process instance (case) will be finished (Van Dongen, Crooy, & Van Der Aalst, 2008); Van Der Aalst et al. presented a method for generating transition system from an event log which is used for time prediction of a case (Van Der Aalst, Schonenberg, & Song, 2011). The approaches are suitable for time prediction, but they are specialised for this specific task and do not take into account additional data attributes.

Alternatively, temporal information can also be used for anomaly detection, such as the one presented in (Ping, Chen, Chen, & Howboldt, 2010). The approach builds Bayesian networks with data about event sequences and their temporal probabilities as additional nodes. The approach is specific to temporal anomalies and does not provide insight how to detect general anomalies. There are additional algorithms for general anomaly detection in BP event logs (Bezerra & Wainer, 2013). The drawback of these algorithms is that they deal with sequences in activities and ignore additional data attributes.

BPA using process mining and the data associated with BPs is a rather new area. De Leoni et al. in (De Leoni, Munoz-Gama, Carmona, & Van Der Aalst, 2014) presented which can be used to create multi-perspective models. De Leoni et al. have also presented an approach which detects data flow from an event log and associates it with a BP model (De Leoni & Van Der Aalst, 2013). Prediction models have been applied to process mining before, but most of the research focuses on the discovery of BP models using prediction models, e.g. Markov chains were used in methods presented in (Cook & Wolf, 1998; Ferreira, Zacarias, Malheiros, & Ferreira, 2007).

Prediction of BP properties by using prediction models built from event logs has been researched before. Most of the research deals with the prediction of a specific parameter, such as duration. In (Verenich, Dumas, La Rosa, Maggi, & Di Francescomarino, 2016) authors present an approach which builds an SVM prediction model and regression models to optimise process flow to eliminate over processing, i.e. activities that are redundant and has no impact on the final result of the process. In (Folino, Guarascio, & Pontieri, 2014) authors implement a prototype method for prediction by using Predictive Clustering Tree where decision rules are mined. In (Ceci *et al*., 2014) authors exploit sequential pattern mining and use additional information about the activities to train nested prediction models and predict follow-up sequences and process instance duration. Methods using machine learning also exist for BP prediction. In (Polato, Sperduti, Burattin, & De Leoni, 2014) authors present a method which calculates the likelihood of all the following activities using Naïve Bayes Classifier and uses

support vector regression for remaining time prediction, but does not focus on the correct next activity prediction. In (Rogge-Solti & Kasneci, 2014), authors employ non-Markovian stochastic Petri nets with elapsed time since last observed event to predict the follow-up event durations, but not follow-up activities. In (Tax *et al*., 2017) authors train Long Short-Term Memory neural networks and use them as predictive models.

Table 1.3. Summary of process mining predictive method purpose and analysed features

Reference	Approach	Purpose	Analysed feature
(Van Dongen *et al*., 2008)	Regression equations	Prediction of time, left for a process instance to end	Activity duration
(Van Der Aalst *et al*., 2011)	Transition system	Prediction of summary time duration of the process instance	Summary duration, but applicable to other aggregable attributes
(Ping *et al*., 2010)	Bayesian network with temporal data	Identification of anomaly based on probability.	Event sequence and duration probability
(Bezerra & Wainer, 2013)	Heuristic algorithms	Anomaly detection	Event sequence
(Verenich *et al*., 2016)	Support Vector Machine prediction model	Elimination of over processing	Activities having no impact on the final result
(Folino *et al*., 2014)	Predictive Clustering Tree	Decision rule mining	Data dependency
(Ceci *et al*., 2014)	Sequential pattern mining	Prediction of follow-up sequences and total duration of process instance	Sequence patterns
(Polato *et al*., 2014)	Naïve Bayes Classifier and uses support vector re-gression	Remaining process instance duration prediction	Activity duration

Table 1.3 presents a summary of identified methods for predicting BP execution features. None of them focuses on the domain data that is available in the logs, and they focus on the prediction of aggregable business features, such as duration.

1.2.3.2. Business Process Behaviour Parameter Detection

While the prediction of BP flow is very useful for BP owners, the prediction is not so useful when individual process behavioural rules need to be detected. Process mining techniques assist in this regard because they can provide information on individual parameters of a BP.

There are process mining techniques which, when applied to event logs, can infer how resources are assigned to activities. Ly et al. in their paper (Ly, Rinderle-Ma, Dadam, & Reichert, 2006) presented an approach on how to detect which staff is assigned to which activities in a BP. They treat the assignment problem as inductive learning problem and use decision tree learning approach to derive staff assignment rules. The approach is limited, as it detects simple assignment rules for activities and cannot detect rules where assignment of a resource to an activity depends on the resource of the preceding activity, e.g. if a person accepts a call for a restaurant reservation, it could also be the same person that makes a reservation of the table. Zhengxing and Huilong presented an approach that mines association rules for resource allocations taking into account the ordered correlations between items in the event log (Zhengxing Huang & Huilong, 2011). The authors of (Senderovich, Weidlich, Gal, & Mandelbaum, 2014a) have presented an approach to mine resource scheduling protocols from event logs. The approach assumes that organisations have a protocol that defines the sequence of activities which are done by a specific set of resources, therefore it is possible to know what activities and what resources will be assigned to those activities if the protocols can be detected. Although the results are interesting, the approach is limited to service-based processes.

The above-presented resource assignment mining techniques are limited to single event assignment rules. Sometimes context is important to facilitate more detailed analysis of the behaviour of resources, and for this, some approaches mine resource behaviour from event logs. One of such approaches is presented in (Ferreira & Alves, 2012), which mines communities from an event log to detect resource interactions, such as:

- Handover of work: analysing resources which are assigned to sequences of activities allows inferring related resources, i.e. the ones that possibly interact and affect the flow of the process.
- Working together: analysis of a single class of events allows to detect groups of resources that belong to the same class and perform the same activities. This detection can be used in inferring resource pools for activities or a number of resources that are required for performing an activity. Also, working together implies that resources were working in the same process instance.

The approach presented in (Ferreira & Alves, 2012) presents a method to create two social network graphs, where one is for the handover of work and nodes in the graph represent resources while arcs represent which resource hands over work to which another resource. Each arc is weighted with the count of times that such handover of work was detected in a log. The other social network graph has the same nodes, but the arcs are undirected and represent which resources have worked together in the same case. Clustering techniques are then applied to the graphs to identify groups of resources that form a community to reduce the complexity of the social network graph.

Finally, process mining has also been applied in decision point analysis. Rozinat and Van Der Aalst in (Rozinat & Van Der Aalst, 2006) present a method to extract rules for control flow point in the process model based on data in event logs. The rules are extracted using classification algorithms such as C.45. De Leoni et al. in (De Leoni & Van Der Aalst, 2013) present a method which calculates alignment in BPs to extract data flow rules between activities. Authors of (Liu et al., 2012) propose to simulate discovered models for use in decision support. Also, process mining has been applied to domain-specific decision analysis as in (Samo, Dewandono, Ahmad, Naufal, & Sinaga, 2015) where an approach is presented which uses process mining and association rule learning for fraud detection.

1.3. Business Process Simulation

Simulation is a process of creating a model of a real-life problem in order to imitate its behaviour without affecting the real-life. Simulation is a process of creating a model which experimentally predicts the behaviour of a real system for the purpose of designing the system or modifying the behaviour. It is not always possible or feasible to make tests in a real environment. Therefore simulation is the perfect way to save costs and test what-if scenarios (Jansen-Vullers & Netjes, 2006).

Simulation is distinct from animation because it imitates the behaviour of an object and does not focus on the animation of a graphical model. By executing a single process instance or a batch of instances, the simulation can help in detecting potential model errors, identify bottlenecks or resource overload. Based on an analysed problem, a simulation can be either deterministic or stochastic (Waddington, 1974). Deterministic simulation's output is solely dependent on the input parameters, and if the same input is fed into the simulation, the output is always the same. Stochastic simulation, on the other hand, contains indeterministic elements and stochastic input, thereby the output of the simulation

is non-deterministic. Stochastic simulation is sometimes also referred to as Monte Carlo simulation due to the random outcome of the simulation.

A different classification of simulation execution is based on how the model behaves in time. There can be either continuous or discrete-event simulation. Continuous simulation, sometimes referred to as system dynamics, is used when the time of flow is important, i.e. in this simulation, time is a continuous function, and all elements of the simulation model can be modelled as functions taking time and state as parameters. This simulation is expensive to create because the elements of the simulation model are mathematical functions. Therefore the creation of such model is a difficult process which requires high skill-set. Discrete event simulation, in contrast to the continuous simulation, pays less attention to time. In discrete event simulation, the time flows in ticks, i.e. the time moves forwards when an event occurs, and the model does not pay attention to what is happening in-between events because at that time the state of the simulation model does not change.

The elements of the discrete event simulation model do not take time as one of the parameters but change the state of the simulation model once activated, and an event in the simulation is a change of the state. BPs are usually modelled with modelling notations based on Petri Nets. Therefore it is clear that discrete event simulation is the best approach to facilitate the simulation of BPs (Hlupic & Robinson, 1998).

A special case of simulation is an agent-based simulation (Szimanski, Ralha, Wagner, & Ferreira, 2013). In this type of simulation, the elements of the model are independent agents that communicate in-between to imitate some domain behaviour. This type of simulation is best applied for simulating multiple independent systems, such as human interactions, process orchestration and others.

BP simulation, as a process, has not changed for the past 20 years (c.f. (Hlupic & Robinson, 1998) (Van Der Aalst, 2015)) - what has changed is how the simulation models are created, how the data for simulation models is collected, or the simulation result application.

1.3.1. Simulation Models

1.3.2.1. Discrete Event Simulation Models

Most of the BPS approaches and tools rely on DES (Jansen-Vullers & Netjes, 2006; Vasilecas, Kalibatiene, & Lavbič, 2016). By default, DES models require the following parameters (Morgan, Banks, & Carson, 1984):

– *Entities*. They are the elements of the models that modify the state of the simulation and interact with other *entities*. They can be either permanent

and stay throughout simulation or appear only temporarily, e.g. during subprocess execution. In BPS, the *Entities* refer to activities, events and other visible model elements.
- *Resources*. A special case entities that are consumed or occupied by other entities to perform some action.
- *Activities*. They are functions that resources perform when consumed or occupied by an entity. The activities can have a duration (static or a distribution of some form). While entities in BPS are reflected by model elements, the activities are the actual behaviour that these elements perform. Usually, BP models do not define this behaviour, and this is added when a BP model is transformed to a BPS model.
- *Attributes*. They identify entities and define their characteristics. In BPS, they are the names
- *Variables*. They reflect the characteristics of the system as a whole and are used to describe the state of the process. The variables are not associated with entities, although the entities might modify the variables.
- *Events.* They are facts that occur at a given time during simulation, and their source is a change in simulation state. The events can be either internal and occur during the execution of a process or external and occur outside of the process. The events are what forces the flow of time during the simulation.

1.3.2.2. Business Process Simulation Models

Standard BP modelling languages pay attention to how process control flow is performed but do not care about the actual behaviour. Most widely used BP modelling languages have model elements to describe the following aspects of BPs (Heidari, Loucopoulos, & Brazier, 2013):

- Behavioural aspect, which defines the control-flow of the process, how activities follow each other, what events occur and how they impact the control flow. This behavioural aspect does not contain information on when the activities occur or how long do they take to occur, because that is not in the scope of the modelling languages.
- Functional aspect which identifies the activities that can be executed in the BP, but the activities are not defined in detail; they are only named, with perhaps references to the actual documentation, describing what is done during the activities.
- Informational aspect, which defines data objects, storage locations used, data that is the input and output of activities and others. More complex

elements, such as messages or conversations are also available in some languages.

– Organisational aspect, which is used to describe the resources, executing the activities or participating in them. The organisational aspect is mostly limited to performers of activities. The resources that are consumed or created during the activities are not modelled, or there are no specific elements to model such elements. Also, resource behaviour, such as allocation time, schedules, priorities, are not modelled in those languages.

It is clear that existing modelling languages focus mainly on modelling the static properties of the processes and the actual behaviour, such as what resources are required by activities, how long do they take to complete or what process state parameters are available during the execution, are out of the scope of such modelling languages.

The missing details in the existing models lead to ineffective BPS. Organisations usually have BPs documented, but they are not suitable for BPS because the models have to be modified to facilitate simulation. Therefore, when there is a need to simulate a BP, the existing BP model has to be extended with simulation parameters (Laue & Mueller, 2016).

Only recently a BPS model standard appeared. The standard is called BPSim (Yi & Filippidis, 2013) and it aims to unify the BP Simulation tools and facilitate interoperability between them. The standard is an extension of BPMN and provides a schema for describing simulation parameters by annotating BPMN models with additional parameters (Fig. 1.4).

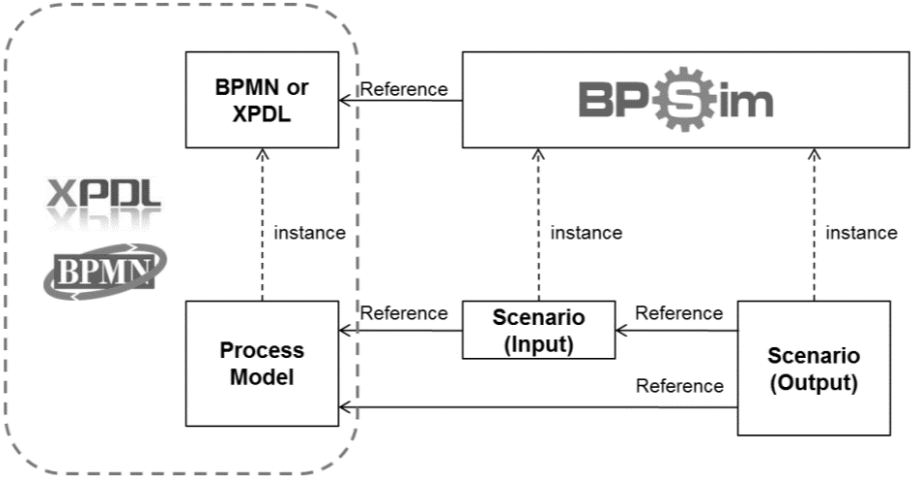


Fig. 1.4. BPSim standard interaction with BPMN and XPDL (Yi & Filippidis, 2013)

The BPSim standard adds additional parameters to BP Model elements, thereby facilitating simulation parameters (Fig. 1.5). The parameters can be the following:

- *TimeParameters* define the time-related behaviour of a process element, e.g. how long an activity takes to complete, the wait time and others.
- *ControlParameters* define the behaviour of control-flow elements such as split nodes, loops and others, e.g. the probability distribution for each sequence flow path, event occurrence probability.
- *ResourceParameters* define parameters of the resources used in BP elements. These parameters only apply to elements, which require resources, e.g. activities, tasks. The parameters define the availability of the resources, their quantity and selection parameters, such as roles.
- *CostParameters* specify the cost of a BP element or its execution. It can be either fixed cost for each occurrence or a cost per unit of time.
- *PropertyParameter* specifies other properties of a BP element which are domain-specific.
- *PriorityParameter* specifies the properties of how the process element interacts with the whole process, e.g. whether it is interruptible or what is the prioritisation of paused elements.

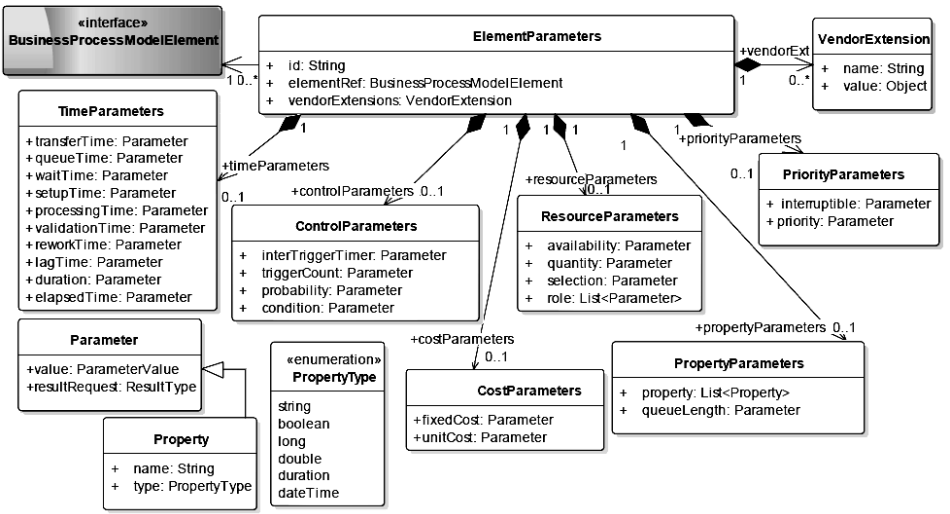


Fig. 1.5. BPSim Element Parameter extension (Yi & Filippidis, 2013)

All parameters in the model have an interval of time for when they are applicable and a value for the time. The value can either be a constant (*String*,

Number, *Float*, *Boolean*, *Duration* or *DateTime*) or a result of distribution. The standard presents a total of 13 distribution possibilities, e.g. Beta, Gamma, Erlang and Uniform.

The standard is supported by multiple BP modelling and simulation tools used in industry, such as Bizagi Modeller (Bizagi, n.d.) or Sparx System Enterprise Architect ("Business Process Simulation (BPSim)," 2017).

The complexity of planning and scheduling is determined by the degree to which activities contend for resources (Fadel, Fox, & Gruninger, 1994) and the final problem with existing BPS solutions is that they have limited details of the process behaviour. The standard business process simulation models cannot represent complex behaviour, such as human interactions in processes (Yi & Filippidis, 2013) or they use simplified general resource behaviour versus the way resources behave (Fadel *et al.*, 1994; Vasilecas, Normantas, Rusinaite, Savickas, & Vysockis, 2016). This forces business analysts to employ non-BPS specific approaches, but general simulation tools such as Arena or Simul8 (Greasley, 2004). The problem with these tools is that they use proprietary modelling languages for generic simulation, making them non-interchangeable with BP management systems (Van Der Aalst, 2015).

Dynamic Business Process Simulation (DBPS) attempts to solve this problem (Vasilecas *et al*., 2015). The DBPS proposes a process modelling and simulation approach, where BPs are interpreted as having no clear control flow and the flow is decided during the execution.

The DBPS takes note of BP terms and elements, thereby keeping a connection with general BP research area. Activities in the DBP are defined using the form $activity = (rule, taskSet)$, which allows maximum possible flexibility of process control flow and $taskSet$ is a pseudo-code definition of how activity modifies the state of the context. Such definition of activities allows to achieve more precise BPS results (Vasilecas, Normantas, *et al*., 2016).

1.3.2. Process Mining Application in Business Process Simulation

For discovering BP behaviour parameters that can be used in the simulation, process mining provides multiple solutions. To name a few of such state of the art methods: Rozinat provides decision rule mining that can be used to predict branching (Rozinat & Van Der Aalst, 2006); Van Dongen et al. in (Van Dongen *et al*., 2008) have presented an approach for predicting activity durations; Senderovich et al. apply process mining for discovering resource scheduling protocols (Senderovich *et al*., 2014c). While there are many possible applications of process mining methods for discovering BP behaviour, their application is limited to discovering parameters that can be used for simulation. Authors of

(Martin, Depaire, & Caris, 2016) performed systematic literature review analysis on what Process Mining methods could be used for discovering data and could be used in BP simulation. On the other hand, there is no reasearch available on the integration of the reviewed approaches in BP simulation, and no experiments were done to investigate the actual value of the methods in BPS.

General simulation model discovery from the event logs has seen only a few applications. In (Ahn, Dunston, Kandil, & Martinez, 2015) authors use a refined alpha algorithm to discover workflow paths and durations which is then used to create a discrete event simulation model, but the simulation model does not contain complex data and decisions. Authors of (Giuseppe, Valerio, Teresa, & Carmela, 2014) present a method to create a risk-based simulation model to perform conformance analysis of a modelled process, and it is focused for that specific task. In (Nagatou & Watanabe, 2015) authors discover performance parameters using well-known process mining techniques and manually, based on them, create a simulation model to analyse what-if scenarios, but the approach is not explained in detail. None of the found approaches tries to create a model which would provide not only activity and their duration simulation, but also data generated in the activities.

The simulation itself can also be applied to improve process mining. In (Szimanski *et al*., 2013), authors use agent-based simulation on BP models to simulate interactions between agents and map the interactions with events in a log. The research uses Hidden Markov Models for modelling event occurrences and maps them to higher level BP. Authors of (Ackermann, Schönig, & Jablonski, 2016) use BP simulation to generate an event log and use the generated event log for mining a different paradigm BP – from declarative process models to well-structured process models. The research focuses on simulating already existing models and does not focus on the creation of detailed simulation models from event logs.

1.4. Conclusions of Chapter 1 and Formulation of Objectives

1. Most of the BP prediction techniques in process mining focus on the state of the process or a single attribute, e.g. time or duration; therefore, their applicability for other, domain-specific, parameters prediction is unknown.
2. Discrete Event Simulation is the most widely applied BP simulation method, although its use is limited due to multiple existing problems – static process models, a single, not widely adopted simulation standard, and manual labour required for model creation.

3. Existing process mining techniques can be applied to discover BPS model parameters, but their practical application has limited research, and their integration and combination are not researched to prove their use in BPS.

Based on the literature review, the following further objectives were formulated to achieve the goal of the dissertation:

1. A method must be proposed which could:
    1.1. discover a business process model from an event log that would take dynamic and stochastic properties of BPs into account;
    1.2. generate a simulation model from discovered business process models.
2. The proposed method must be experimentally evaluated using event logs of multiple BPs with varying degree of complexity.

2

Discovery of Bayes Belief Network and Simulation Model from an Event Log

The improvement of BPs can be performed using multi-perspective analysis. Based on the expected result, the analysis focuses on three different perspectives: the past, the present or the future. The analysis of the past includes the investigation of historical data on how a BP has been executed to gain insights on what could be improved. The analysis of the present focuses on the BPs that are being currently executed to support decision-making activities and to assist in the detection of anomalous activities. Finally, the future analysis focuses on one of two tasks – the prediction of the near-future behaviour of the current process instance or the prediction of process behaviour in the long term for what-if analysis.

This chapter presents a method to create a probabilistic Bayesian Belief Network from an event log and, furthermore, use the belief network for simulation model creation. The method uses historical data of the BP execution to create a belief model which allows gaining insights into the behaviour of the past, identify anomalies for BP instances and to facilitate near-future prediction of the currently running process instances. The generated simulation models can assist business analysts in performing BP simulation and allows what-if analysis. The contents of this chapter are based on previously published content by the author in

(Kalibatiene, Vasilecas, Savickas, Vysockis, & Bobrovs, 2016; Savickas & Vasilecas, 2014, 2015, 2017; Vasilecas, Kalibatiene, *et al*., 2014; Vasilecas *et al*., 2015).

2.1. The General Approach

Two of the main issues that hinder BP analysis is that the existing knowledge available in the ISs is not reused and that the process of modelling the domain is a time-consuming task (see chapter 1). That is where lies the idea of the proposed method: there is a need to extract already-existing knowledge in IS about the process execution and to automate domain modelling. Having achieved the mentioned objectives, the approach would facilitate faster analysis of the BPs the users are interested in.

The method starts with the re-use of existing knowledge. All BPs, when automated and executed in an IS, leave a trace of its execution in some form. How much data is in the traces depends on an IS, e.g. process-aware ISs keep a detailed log of the activities, their performers and other associated domain data, such as locations, customer details, and others. Other systems, which are not process-aware, might store such traces in an inexplicit form, e.g. files, relational database tables and others. In any case, this data is available in some form, but due to the inexplicit, non-standard format, it cannot be extracted in the same manner each time. Therefore, the extraction of such data is a manual task.

The idea to use historical data for BP analysis is not a new one, and it is the basis of the process mining research area (see chapter 1). The proposed method follows the same path as existing process mining methods and ignores the type of the source data. Due to the unpredictability and non-uniformity of the data source, the approach assumes that it is the duty of the user to acquire an event log with the representation on how a BP has executed in the past. The event log is a well-known construct in the process mining research area and is described in more detail in section 1.2.1.

The other problem lies in the domain modelling. The domain model must satisfy the following constraints in order to be usable for analysis:

1. The process is a set of activities executed in a specific sequence to transform input into output. Therefore, the model must represent control flow of the process.
2. The process has input, output and generates data. Therefore, the model must take into account the data occurring in the BP that is stored in an event log.
3. The process has stochastic nature due to human interactions, unknown and unpredictable context (such as weather, clients, third party systems)

and other factors. Therefore, the model must represent this stochastic nature.

4. The process contains feedback mechanisms because behaviour and decisions made at one point in the process might have an impact on behaviour later in the process in a complex or an indirect way. Therefore, the model must represent feedback mechanisms.

The constraints mentioned above eliminate most of the standard BP modelling languages, such as BPMN, UML Activity Diagrams and Petri Nets. It is so due to the fact that even though the standard BP modelling languages can represent control flow and data, they cannot represent stochastic nature and implicit behaviour mechanisms. Their purpose is to provide a static view that represents how the process should behave in a perfect scenario, e.g. to depict how the activities in the BP follow each other via activity and arc elements. The constraints depicted in the model are often overlooked in real-life scenarios, or, in other cases, there are no elements that depict feedback mechanisms between activities not directly linked by arcs. Also, the modelling languages often lack a detailed view of the data that is occurring in the process and is generated in software systems, because it is deemed not useful by the analysts. Another drawback of such models is their limitation to represent the behaviour of the past, e.g. how often and for what reason the splits in the control flow of the process were made.

An alternative to standard BP modelling languages are artificial intelligence models, such as Support Vector Machines (SVM) and Neural Networks (NN). Such models do not satisfy the above-mentioned constraints because they can represent stochastic nature and implicit feedback mechanisms in the BP, but a single model cannot represent control flow and data in a comprehensible and visible manner. They are mathematical, non-graph-based models and this makes it hard to depict BPs in a manner that would allow easy identification of activity sequences or attribute dependencies or explain the reasoning behind the models.

Only the probabilistic, graph-based methods are left. Markov chain could be a suitable candidate, but it assumes that given the past, the future is independent. Therefore Markov chains do not satisfy the 4^{th} constraint above. The final candidate is the Bayesian Belief Network (BBN). It is suitable, because:

- the underlying Directed Acyclic Graph (DAG) could represent the control flow;
- the conditional probability tables represent data used in the process and exhibit:
  - attribute and value space of the data in the process;
  - stochastic nature via probabilities of data occurrence;
  - feedback mechanism using joint probability tables.

The main limitation of the BBN is that the underlying graph is directed and acyclic. Such limitation reduces the control flow depiction capabilities, i.e. the BBN does not permit loops and cycles, but it is offset by the ability to use the belief network to analyse historical data and the capability to:

– see possible event sequences;
– detect anomalous behaviour by inferring the low probability of some execution path;
– predict the near-term future by finding the maximum conditional probability of some execution path.

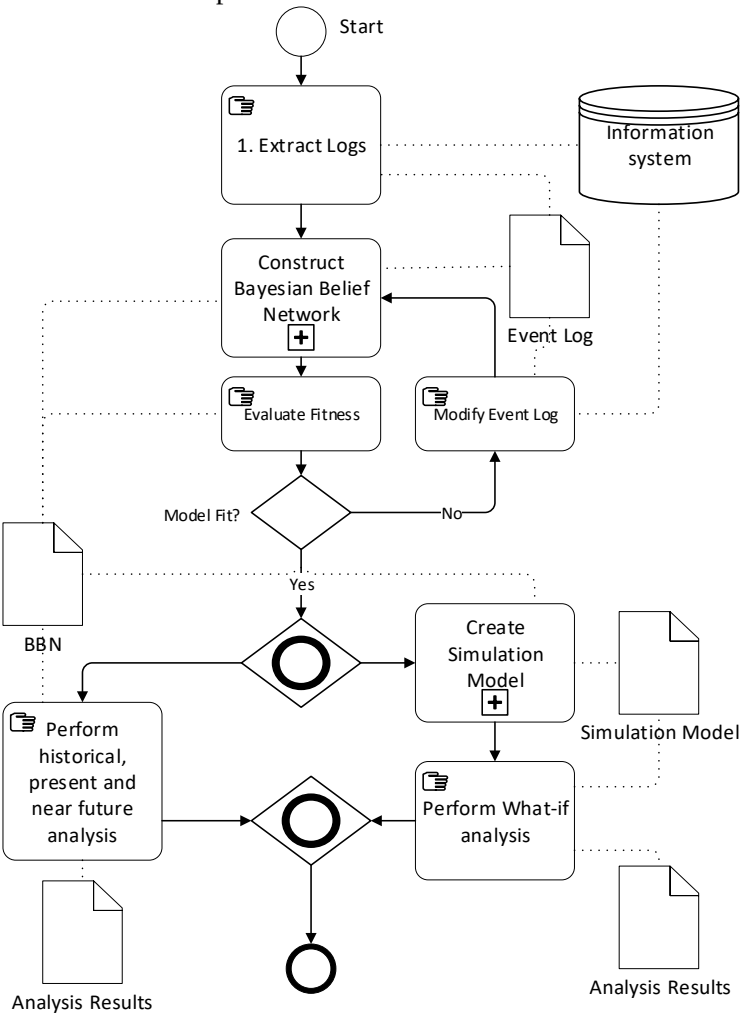


Fig. 2.1. Conceptual Model of the approach

The BBN can also facilitate BP simulation because the underlying model could allow inference of data probabilities and it could be used to generate data or decide what path the process should follow based on the past behaviour.

Finally, the BBN by itself is not suitable for investigating what-if scenarios. It facilitates the inference probabilities for the underlying data and the prediction of the data. The *de facto* approach to analyse what-if scenarios is BP simulation. Therefore it must be the final step in the overall approach.

As the outcome of the deliberation in this section and the ideas, the concept of the method was designed – to use historical data from IS describing historical BP execution to create a BBN and furthermore use it to generate a simulation model to facilitate historical, present and future execution of BPs. The conceptual model of the approach is presented in Fig. 2.1.

The method consists of the following activities:

1. **Name:** Extract Logs
   Input: Information system and the corresponding data storage.
   Activity: An information system is analysed, and an event log depicting historical execution of a BP is extracted. The activity is a manual task and is dependent on the system.
   Output: Event log.
2. **Name:** Construct Bayesian Belief Network
   Input: Event Log
   Activity: Data in the event log is transformed into Bayes Belief Network.
   Output: Bayes Belief Network
3. **Name:** Evaluate Fitness
   Input: Bayes Belief Network
   Activity: The Bayes Belief Network is evaluated whether it fits the event log, i.e. whether the graph is correct, whether prediction rate and probability inference are sufficient for analysis, etc.
   Output: Fitness
4 **Name:** Modify Event Log
   Input: Event Log, Information System
   Activity: Event Log is modified to improve the fitness of the discovered Bayes Belief Network. The activity is manual and might remove attributes from the event log, might select different activities, might modify data (cluster values, etc.).
   Output: Event Log
5. **Name:** Create Simulation Model
   Input: Bayes Belief Network
   Activity: The belief network is used to create an initial simulation model which imitates the behaviour of the event log.
   Output: Simulation model

6. **Name:** Perform historical, present and near-future analysis
   Input: Bayes Belief Network
   Activity: The analyst uses the model to:
   check probabilities of the historical behaviour, e.g. how often an activity occurred given some client information;
   infer the probability of the currently executed BP instance to detect anomalous behaviour;
   predict BP behaviour of the current instance given the current context;
   Output: Analysis results for the given task.
7. **Name:** Perform what-if analysis
   Input: Simulation model
   Activity: The simulation model created from the belief network is used to modify the process behaviour and assess the impact of the change.
   Output: Analysis result for the given task.

The following sections in detail describe the process of creating each of the outputs in the stages.

2.2. Event Log and Process Instance State

The data in IS is not uniform. Therefore an intermediate form of BP historical execution data format is needed. The process mining research area has long used event logs (Van Der Aalst, De Medeiros, & Weijters, 2005a) to describe BP execution history and all of the process mining approaches use the event logs as the main input. It is for the same two reasons – to have a unified form for data and to have data which describes the historical execution of the BP – that event log is chosen as the input for the method (Fig. 2.2). To eliminate the need for a specific file format, we introduce an event log definition which abstracts away the actual implementation of the event log.

The event log contains process instances (or cases, or traces) where each instance is defined by a set of events that have occurred during the process instance. Each event, furthermore, is described by one more or more data attributes. The method uses the definition based on research by Van Dongen et al. in (Van Dongen *et al*., 2008). The original definition did not have individual attribute values of the events because their work was focused on totals or an attributes values in all events to predict some parameter or a case. In this case, however, this is not suitable. We need individual attribute and value pairs for them to be usable during the creation of the BBN. Also, the ordering of events is not used in the method above, but it is needed to model the DAG, which is a component of the BBN (see section 2.3).

```
<trace>
    <string key="concept:name" value="1632"/>
    <string key="description" value="Simulated process instance"/>
    <event>
        <string key="org:resource" value="40-49"/>
        <date   key="time:timestamp" value="1970-01-01T04:47:08.000+01:00"/>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:name" value="incoming claim"/>
    </event>
    <event>
        <string key="org:resource" value="Center Klaipeda"/>
        <date   key="time:timestamp" value="1970-01-01T05:53:52.000+01:00"/>
        <string key="lifecycle:transition" value="complete"/>
        <string key="location" value="Klaipeda"/>
        <string key="concept:name" value="B check information"/>
    </event>
    <event>
        <string key="org:resource" value="Center Vilnius"/>
        <date   key="time:timestamp" value="1970-01-01T06:04:04.000+01:00"/>
        <string key="lifecycle:transition" value="B register claim"/>
        <string key="concept:name" value="complete"/>
    </event>
</trace>
```

Fig. 2.2. Example of an Event Log in XES format

Definition 4. *An event log over a set of activities A and time domain TD is defined as* $L_{A,TD} = (E, I, N, M, V, \mu, \alpha, \gamma, \beta, >, \succ, name)$*, where:*

- E *is a finite set of events;*
- I *is a finite set of cases (process instances);*
- N *is a finite set of attribute names;*
- V *is a value space of attributes;*
- $M: N \times V$ *is a finite set of attributes;*
- $\mu: E \to M$ *is a function assigning each event with attributes and their values;*
- $\alpha: E \to A$ *is a function assigning each event to an activity;*
- $\gamma: E \to TD$ *is a function assigning each event to a timestamp;*
- $\beta: E \to I$ *is a surjective function assigning each event to a case;*
- $name: E \to N$ *is a function identifying the name of an event and* $name(e) = v: (v \in V, n \in N: (v, n) \in \mu(e) \wedge n = "name")$*;*
- $> \subseteq E \times E$ *is the succession relation, which imposes a direct ordering of the events in E;*
- $\succ \subseteq >^+$ *is the succession relation, which imposes a total ordering of the events in E.*

Graph-based models discovered from event logs are not fault-tolerant and could be underfitting. Therefore, using such approaches might sometimes lead to erroneous predicted behaviour. For example, if there is a graph $\{\{a,b\},\{b,c\},\{b,d\},\{c,d\}\}$, it permits a sequence of $\{a,b,d,c\}$ to occur, which might not exist in the original event log. Therefore, a state during any process analysis should be tracked and the possible execution or inference paths should be constrained.

Authors of (Van Der Aalst *et al*., 2011) introduced a labelled state transition system which was used to predict the duration of a process instance. This transition system keeps track of a process state and allows the identification all possible execution paths for a given partial traces of an event log.

Definition 5. *Given a state representation function l^{state} and an event representation function l^{event}, a labelled transition system is defined as $TS = (Y, E, T)$ where Y= $\{l^{state}\left(hd^k(\sigma)\right) | \sigma \in L \wedge 0 \leq k \leq |\sigma|$ is the state space and $hd^k(\sigma)$ is a "head" of event sequence in a trace of first k elements. $E = \{l^{event}(\sigma(k)) | \sigma \in L \wedge 1 \leq k \leq |\sigma|\}$ is the set of events labels, where $X = \left\{l^{state}\left(hd^k(\sigma)\right), l^{event}\left(\sigma(k+1)\right), l^{state}\left(hd^{k+1}(\sigma)\right), l^{event}\middle| \sigma \in L \wedge 0 \leq k \leq |\sigma|\right\}$ is the transition relation. $Y^{start} = \{l^{state}(\langle\ \rangle)\}$ is the singleton of initial states and $Y^{end} = \{l^{state}(\sigma) | \sigma \in L\}$ is the set of final states.*

The definitions of the trace state l^{state} and the event state l^{event} are left unused and undefined in the original work, therefore, for the method these definitions are introduced. The trace state l^{state} is used in tracking the state of the current process instance and the event state l^{event} is used to track the data that describes the event in an instance of the BP.

Event state l^{event} describes attributes and their values that belong to the specific occurrence of an event in a trace, therefore it can be defined using the definition of μ:

Definition 6. *Event state is defined as $l^{event}(e) = \mu(e), e \in E, \mu(e) \in M$ and it describes attributes and their values of a specific event.*

A state of a trace is a collection of event states; therefore, it can be defined as:

Definition 7. *Trace state for a partial trace is represented as a set of event states $l^{state}(\sigma) = \left\{e, M_e, e_{previous} \middle| e \in E \wedge \forall \alpha(e) = t \wedge t \in \sigma, M_e \in M \wedge \mu(e) = M_e, e_{previous} > e\right\}$.*

The use of event log and process state definitions allows operation using formal methods and abstractions which later on can be implemented in any system. These definitions are the basis for constructing the BBN and simulation models and, furthermore, facilitate the modelling of a process state for the hypothesis in the inference of the probability of the simulation state (see section 2.3.3).

2.3. Bayesian Belief Network Construction

Bayes Belief Network is a model that can depict causality between data. The BBN consists of two main parts – directed acyclic graph (DAG) and conditional probability tables (CPT). They are two separate components where DAG is used for depicting event conditional independence and CPTs depict the probability of data in a specific node (Fig. 2.3).

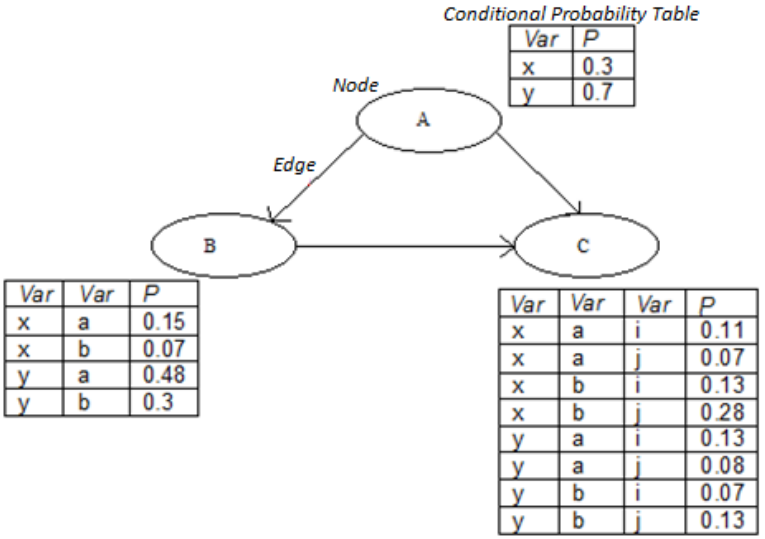


Fig. 2.3. Illustration of Bayes Belief Network

There is only a single alternative method that can create a Bayesian Belief Network from an event log (Sutrisnowati, Bae, Park, & Ha, 2013). It is a general method for creating Bayes belief network. The method does not take into account the underlying control flow of the BP but rather attempts to discover it by analysing dependency between events in the log. It is for this reason that the constructed Directed Acyclic Graph is not usable to depict the sequencing of the events. The method, furthermore, is not tested for anomalies or the current state probability inference; therefore, its usability is not proven. The following sections

present a novel method for building a BBN from an event log (Fig 2.4), as the underlying graph is built to specify the sequences of events and the conditional probability tables provide a way to infer the probability of occurrence of an event or the data probability of data describing a specific event.

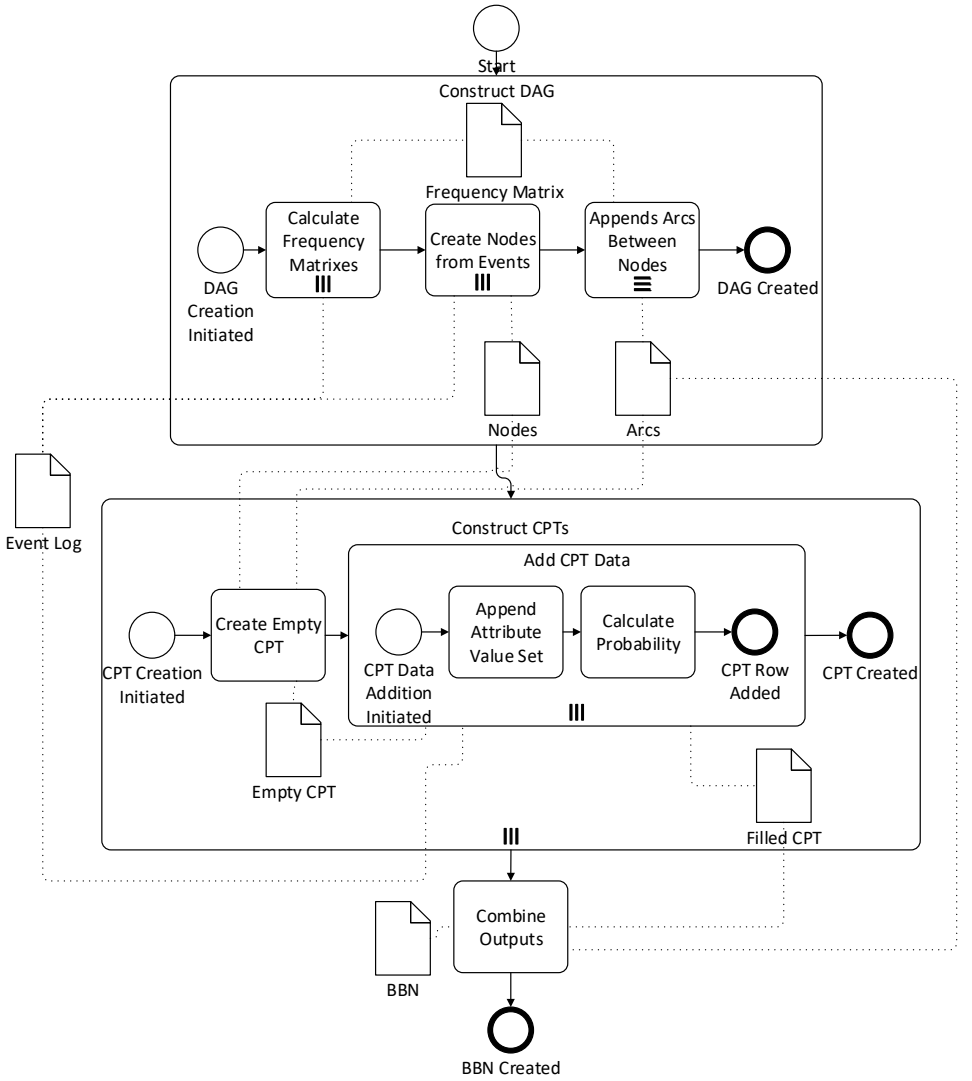


Fig. 2.4. The process model for discovery of Bayes Belief Network

2.3.1. Directed Acyclic Graph Extraction from an Event Log

The directed acyclic graph is one of two core components of the Bayesian Belief Networks. Since the graph is directed, it can represent the sequences of activities in the BP. DAGs cannot fully represent a BP due to a vital difference from the standard BP modelling languages – the DAGs cannot have cycles which are a common component in BPs. On the other hand, it is possible to model the cycles using different approaches, e.g. identification of a loop and transformation into a junction tree (Jensen, 1996). Another alternative is to calculate beliefs by allowing the cycles, although undirected, using belief propagation for inference (Murphy, Weiss, & Jordan, 1999). Bayesian Belief Network requires directed acyclic graph and the usual approaches for discovering BP models are unsuitable because the discovered models are non-acyclic (see section 1.2.1 and (Augusto *et al*., 2017)). These methods allow edges in the graph to exist that could form a loop.

It is clear that the graph must depict control flow of the process to make the discovered DAG semantically similar to the original BP and be usable by the BBN. In other words, the arcs have to go between events that can follow each other in the log, but the graph must not have loops. Causal Net acts in a similar fashion (Weijters & Ribeiro, 2011). It evaluates the frequency of events following each other in the log to build the graph, but the resulting discovered graph is not acyclic. Therefore it is not suitable for BBN. For this reason, the proposed method uses similar frequency calculations, but the graph building algorithm differs in order to create an acyclic graph.

The loops in the graphs, discovered from events logs using existing Process Mining methods, have the following causes:

- The BP contains a loop, and it is extracted as such from the event log.
- The BP contains parallel activities which are extracted as sequences from the event log, thereby forming a loop. This extraction renders the model semantically incorrect and modelled activities become causally dependent, while in reality, they are independent.

A process instance loop is defined as a set of events that are ordered in sequence, but the first and last events in the set appear in the log more than once and they belong to the same trace:

Definition 8. *A process instance loop in an event log L is a set of events* $Z=\{p_0,p_1,..,p_{i-1},p_i\}$ *where:*

- *Z is a finite set of events;*
- $\forall z \in Z: z \in E$ *all events in the set belong to the event log;*
- $\forall z_k, z_j \in Z : z_k \succ z_j$ *, where* $k > j, k > 0, j > 0, k < i, j < i$ *, all events inside the loop are ordered;*

– $z_0 \succ z_i \& z_i \succ z_0$, *first and last elements follow each other in the trace;*
– $\forall z \in Z, E \rightarrow i \in I$, *all events in a loop belong to the same trace.*

For example, there may be traces *ABCD*, *ACBD* and *ABBCD*. The first two traces do now allow for the loop to exist, but standard process mining tools would extract it in a way to allow a loop (Fig. 2.5a). The correct way would be to model them as independent activities (Fig. 2.5b). For the third trace, the process allows the loop (Fig. 2.5c), but this construct cannot exist in the directed acyclic graph.

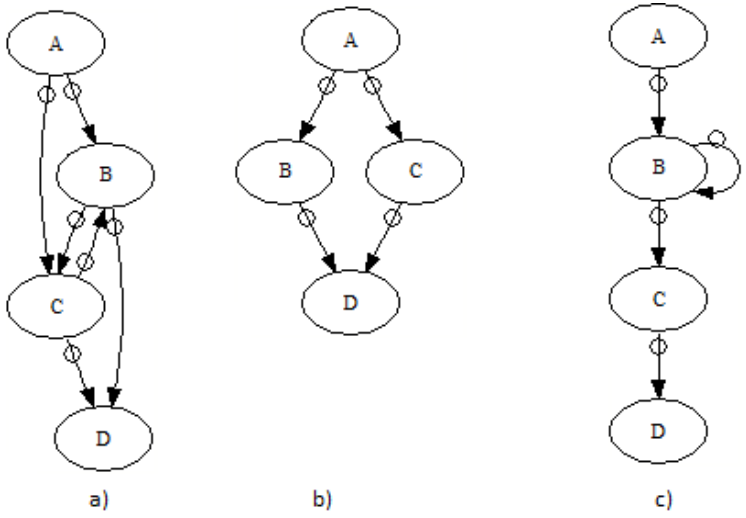


Fig. 2.5. Loop discovery types: a) Incorrect, and b) correct graphs for traces "ABCD" and "ACBD"; c) loop of trace "ABBCD" (Vasilecas, Savickas, *et al*., 2014).

An incorrect model level loop is defined as a set of nodes that are ordered in a sequence and its first and last elements and their corresponding events never form a loop in any trace in an event log.

As it can be seen, the goal is to transform an event log into a graph, where events follow each other in a way that would not form a loop. Having identified the problem, it is clear that there is a need to identify the loops and eliminate them.

First of all, the directed acyclic graph is defined as follows:

Definition 9. *Directed acyclic graph over event log* L *is defined as* $T_L = (F, D, M, \delta)$, *where:*
– $F = \{f \in E: \nexists e \in E e \succ f \& f \succ e\}$ *is a set of nodes for each subset of events not forming a cycle;*

– $D = \{F \times F: f_i, f_j \in F, f_i > f_j\}$ *is a set of edges connecting nodes, whose representative events directly follow each other;*
– M *is a finite set of attributes;*
– $\delta: M \to F$ *is a surjective function assigning each attribute to a node.*

The frequency of event followings is calculated in a matrix. The calculations are done by iterating through traces, and each trace is processed to identify how events behave both in the same trace and in different traces, i.e. whether they directly follow each other, form loops, eventually follows each other or never appear together. The sequence matrix is defined as follows:

Definition 10. *A sequence matrix is a set* $M_f = (N, \partial, \vartheta, \Omega, \partial, \sigma)$, *where:*
– $N \in E \times E$ *is a set of event pairs;*
– $\partial: E \times E \to \mathbb{N}^0$ *where* $\partial(e_i, e_j) = |\{\exists e_i, e_j (e_i \in E \ \wedge \ e_j \in E \wedge e_i \neq e_j \wedge e_i > e_j)\}|$ *is a function assigning each event pair a count of times when the event* e_i *directly follows event* e_j*;*
– $\vartheta: E \times E \to \mathbb{N}^0$ *where* $\vartheta(e_i, e_j) = |\{\exists e_i, e_j (e_i \in E \ \wedge \ e_j \in E \wedge e_i \succ e_j)\}|$ *is a function assigning each event pair a count of times when the event* e_i *follows event* e_j *somewhere in an instance of a process;*
– $\Omega: E \times E \to \mathbb{N}^0$ *where* $\Omega(e_i, e_j) = |\{\exists e_i, e_j (e_i \in E \ \wedge \ e_j \in E \wedge e_i \succ e_j \wedge e_j \succ e_i)\}|$ *is a function assigning each event pair a count of times when the event* e_i *is followed by event* e_j *and follows the same event somewhere in an instance of a process.;*
– $\rho: E \times E \to \{0,1\}$ *where* $\rho(e_i, e_j) = \{\exists e_i, e_j [e_i \in E \ \wedge e_j \in E \wedge (\vartheta(e_i, e_j) < \tau \ \wedge \ \vartheta(e_j, e_i) < \tau \to \rho = 0) \wedge (\vartheta(e_i, e_j) > \tau \ \wedge \ \vartheta(e_j, e_i) > \tau \to \rho = 1]\}$ *is a function declaring whether the events ever occur in the same case;*
– $\sigma: E \times E \to \{0,1\}$ *where* $\sigma(e_j, e_i) = \{\exists e_i \exists e_j \left[e_i \in E \ \wedge e_j \in E \wedge \left((\vartheta(e_i, e_j) > \tau \ \wedge \vartheta(e_j, e_i) < \tau) \vee \left((e_i, e_j) > \tau \ \wedge \partial(e_j, e_i) < \partial(e_i, e_j) * 0.1\right) \to \sigma = 1\right)\right]\}$ *is a function declaring whether event can be connected in a graph.*

In the matrix calculations, the argument τ is used to denote a noise threshold. In best case scenario, where logs contain no noise, this threshold could be zero value. In extensive testing, best results were reached when τ was selected to be alpha level of matrix with value 0.1, but the threshold could be set on case by case basis. In contrast with (Weijters & Ribeiro, 2011), the discovered graph should not have cycles. For this, the tree is formed iteratively and, on each addition, it is

checked whether the addition would form a cycle. If it does, a nearest previous node in the graph, which does not form a graph, is selected. This reflects the idea, that the node nearest in the chain before the cycle is the one that causes the loop.

Having the matrix as defined above, the event log can be transformed into a Directed Acyclic Graph with the following algorithm:

```
For each trace in traces
 For each event in trace
  If Graph.NodeCount = 0
    CreateRootNode(event.previous)
  If Suitable(event.previous, event)
    Graph.AppendNode(event,event.previous)
  Else
    Let x = Graph.NearestSuitable(event,trace)
    Graph.AppendNode(x,event)

Function Suitable(PreviousEvent,Event)
 If
 ρ(event.previous,event)>τ &
 σ(event.previous,event)>τ &
 ¬Graph.CreatesLoop(event,event.previous)
  Return True
Else
  Return False
```

The algorithm iterates through each trace in the event log and each of the events in the traces. Initially, the graph is empty. Therefore the first event of the first trace is used as the root node of the graph. Afterwards, for each event in the trace, the algorithm checks whether the event can follow the preceding event using the frequency matrix. If it is deemed that the arc between the events is suitable, the event is added as a node, and the arc is created with the preceding event. If the node would be not suitable to be connected (would form a loop or fails the threshold value), the function *NearestSuitable* would identify any other event in the graph that is suitable for the arc. Once found, the arc is created between the current event's node and the suitable node. If for any reason, there is no suitable node found in the existing events, the returned event from the *NearestSutiable* is none. In that case, the *AppendNode* function creates a new root node *Start_event*, connects the current root node as its child event, and connects the current event's node with an arc to the *Start_event* node.

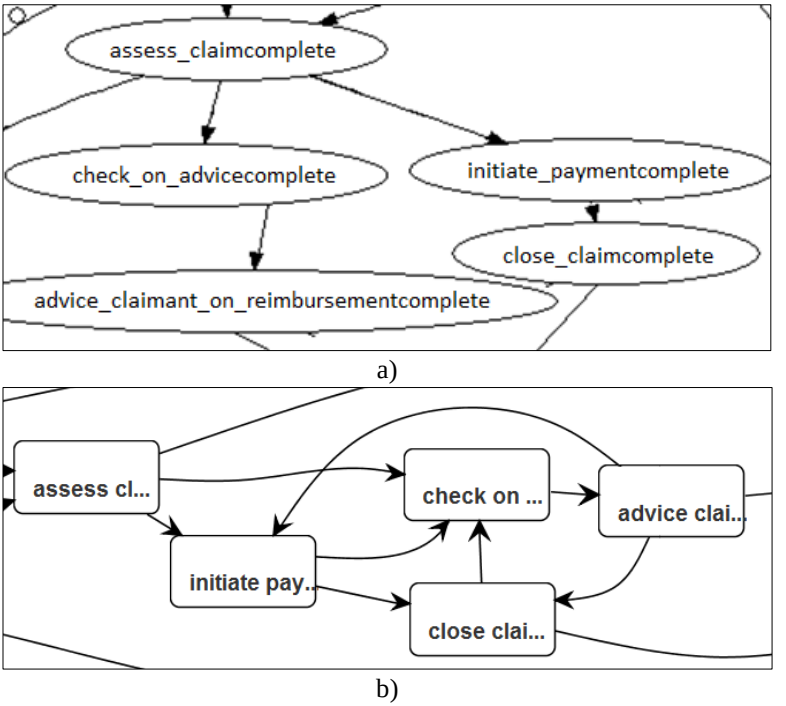


Fig. 2.6. Exemplary fragments of the graph discovery: a) Directed Acyclic Graph with independent events; b) the same events with alternative result of Heuristic Miner

Fig. 2.6 presents an example of the behaviour of the algorithm. The resulting graph of the algorithm is compared with the resulting graph of the Heuristic Miner. The Heuristic Miner discovers a graph with a possible loop $\{initiate\ payment, check\ on\ advice, advice\ claimant, intiate\ payment\}$, whereas the proposed algorithm prevents the cycle. It does so by identifying that the events *check on advice* and *initiate payment* are independent and therefore should be executed in independent paths of the process. Moreover, the {*assess claim, check on advice, advice claimant on reimbursement}* pathway is independent of *{initiate payment, close claim*}, therefore these pathways should not be interleaved, even if in the event log the corresponding events are interleaved in time.

The resulting Directed Acyclic Graph is a representation of a BP model and how the events in the process are time and causally related. The resulting graph is denoted as a tuple $G(E, L)$, where E is a set of nodes representing events and L are arcs connecting events to each other.

2.3.2. Conditional Probability Table Construction

The conditional probability table (CPT) is another core element of the Bayesian Belief Networks. They contain information on how data attributes of the nodes are related to each other. The relation is presented as marginal probability values of $P(X|Y)$, where X is the variable of one node and Y is the variable of a connected node. In the context of event logs, the CPT can be defined as:

Definition 11. *A Conditional Probability Table of an event is defined as a tuple* $\theta(A_e, V_e, \omega)$*, where:*

- $A_e = \{x: \exists e_i \exists e_j [e_j \in E \wedge e_i \in E \wedge e_j \prec e_i \wedge \beta(e_i) = \beta(e_j) \wedge x = \mu(e_j)]\}$ *is a set of all possible values for each attribute of previous nodes in the graph;*
- $V_e = \{x: \exists e_j [e_j \in E \wedge x = \mu(e_j)]\}$ *is a set of attributes and their values belonging to the main node of the probability table;*
- ω *is a probability function* $P(V_e|A_e)$ *assigning conditional probability for each possible attribute value of the main node related to attribute value set of parent nodes.*

The conditional probability table is constructed for each node in the discovered directed acyclic graph. Since a node in the graph depicts an event in a BP, the probability of an event's attributes can be calculated by iterating through the log. Each parent node has a set of attributes and values which, it is assumed, impact the occurrence of values of a child node following the parent. A real-life example of this impact could be as follows: for a decree to approve a final thesis to occur, the topic must be given for a student in university at *study year 3* and the *year* of the final defence of the thesis must always be equal to or higher to the said value of the *study year*. The rule is broken only when an event to pause studies also occurs. Therefore, the value of attribute *study year* is directly dependent and caused by previous event attribute's *study year* value and based on whether any pausing decree (*Occurred* or *Previous* attribute value *1* on any pausing event) has occurred. Since contextual data for the process is usually limited to what is available in the system, no distinction is made on the data – all data attributes available in the log are used. To eliminate data attributes with no impact or to reduce the dimensions of the CPTs, null-hypothesis check might be performed, but this is left for further research.

For each attribute and its possible values, a probability for it to occur together with each possible value set of parent nodes is calculated and stored. Parent node events might not always occur together with the event of the CPT, or the event of the CPT might not always occur when the parent node's event occurs. Therefore, the method uses only those process cases where only the event of the

CPT and related events (connected nodes in the graph) have occurred. When not all of the events occur in the same case, there must be a value depicting a non-occurrence. Such non-occurrence is denoted as an empty element in value sets A_e and V_e. Having data attributes and a graph allows the modelling of dependence between events and related data. For prediction, sequencing must also be taken into account. BPs can have complex control flows (Russell *et al*., 2006) and these have to be taken into account to for correct prediction. For *directly-follows* prediction, only two attributes are required – *occurred* and *previous.* The two attributes – *Previous* and *Occurred* – are stored in the CPT for each parent node. The attribute *Previous* can have values of 0 or 1 and depicts whether the event has previously occurred in the same case the current event has occurred. The attribute *Occurred* can also have values of 0 or 1 and depicts whether the event has occurred in the same process case as the event of the CPT. Other complex control flows could also be added, but the extraction of such control flow is out of scope of this thesis. Finally, the CPT can be created using the following algorithm:

For each node $e \in E$ **in** graph G:
  Let $N = e \cup \{e_j : \exists e_j \wedge e_j \in E \wedge \forall (e_j, e) \in L\}$;
  Let $T = \{t : \exists c[c \in C \wedge \exists n \in N \wedge c = \beta(n) \wedge t = c]\})$;
  For each case c **in** T:
    Let $S = \{a : \exists e[e \in E \wedge \beta(e) = c \wedge a = (e, \mu(e))]\}$
    For each node n_l in N
      If $\nexists\left(n_l, \{(a_i, v_j)\}\right) \in S$
        Update set S **with** values -1 for all attributes of the nodes
        Update the S **with** $(occurred, -1, n_l)$
        Update the S **with** $(previous, -1, n_l)$
      Else
        Add $(occurred, 1, n_l)$ **to** the set S
        If node $n \in N : e > n$
          Add $(previous, 1, n)$ **to** the S
        Else
          Add $(previous, 0, n)$ **to** the S
      If there is an element in θ for the attribute-value set of the case
        Update the ω value
      Else
        Set probability to $\frac{1}{count(T)}$

Based on Definition 11, all calculated conditional probability tables have minimal required size, because A_e and V_e contain only those values that have occurred at least once in an event log. Therefore, for example, for any integer data type, the value ranges are a subset of the integer set. Furthermore, the attributes in an event log might contain many different values or data types, therefore it is mandatory to make some kind of aggregation or transformation of the value ranges, e.g. transform strings into unique ids, use clustering algorithms to reduce value range of floating point data type and so on. This is by itself a topic of statistic and data preparation, therefore transformations or rounding techniques of variable values are omitted in this thesis. The values of the events are assumed to be discrete values either as extracted from the data source or due to pre-processing of the event log .

2.3.3. Business Process Execution Inference Using Bayes Belief Network

BPs, once automated in an IS, have a controlled work-flow. During this work-flow, performers of the process generate data with regards to the process execution, such as location, organisational resource or other domain-specific data such as student group, faculty and similar. This data, once taken as a whole and used in a belief network facilitates detection of causality between events or between data parameters.

The usual approach for analysing BP execution is to use statistical data, such as *averages*, *maximums*, *minimums*, *sums*, *frequencies* and others. While this does provide a means to infer how the process behaves, it could be superficial, because it might not take into account conditional dependencies between data describing the events. To solve this, a Bayesian inference could be used for decision support, because it provides reasonable expectations.

Bayesian inference derives the posterior probability using well-known Bayesian inference formula (Darwiche, 2008):

$$P(H|D) = \frac{P(D|H) \times P(H)}{P(D)}. \tag{2.1}$$

In here, $P(E|H)$ is the posterior probability of a hypothesis H based on evidence D and it is a consequence of two antecedents – the prior probability $P(H)$ and a likelihood $P(D|H)$ with a marginal likelihood $P(D)$.

For BPs, the hypothesis is any set of event attributes and values whose probability we would like to infer. For example, for a question *"what is the probability of the **claim status** to be **declined**?", claim* is the event, *status* is the attribute and *declined* is the value. Even though the BPs can drift and mutate over

time, the possible choices for the hypothesis are limited to those attributes and value pairs which have been seen before in the trace.

Definition 12. *Hypothesis of a BP is defined as* $H_t \in N \times V, \exists e_i \in E, h \in H_t : h \in \mu(e_i)$ *– a set of event's attributes and value pairs which have been observed in the past in the log.*

The hypothesis is not limited to a single $\{e, m\}$ tuple, because there can be multiple events occurring in the same process instance.

As the hypothesis contains multiple possible elements, the prior probability is calculated as a product of probabilities for each of the attribute values to occur with no conditions, i.e.

$$P(H) = \prod_i P(h_i) = \prod_i \omega(h_i). \quad (2.2)$$

In the standard statistical methods, only the number of times when $H|E$ occurred would be used for inference, but this is not really useful for decision support, because it does not take into account the marginal likelihood and only shows a number of times it has been seen regardless of the likelihood of each of the parameters. In other words, the Bayesian Inference not only shows how often this hypothesis has been seen before, but also how likely is it to be seen given the current evidence. Therefore, for inference, we need to use the evidence likelihood. We assume that the inference is done in the context of a single process instance, therefore the evidence is the current state of the trace of the process – l^{state}.

Given a partial trace σ and the current state of the process $l^{state}(\sigma)$, the evidence for a hypothesis is defined as a set of events that have occurred in the current partial trace and their attribute value pairs, i.e. $D = \{(e_i, m_i) | e_i \in \sigma, (e_i, m_i) \in E \times M, e_i \in \sigma, m_i \in \mu(e_i)\}$

Given the definition of the evidence, marginal likelihood can be calculated as a sum of all probabilities for the evidence to occur with the subsets of a hypothesis, i.e.

$$P(D) = \sum_i P(D|h_i) \times P(h_i) = \sum_i \left(\prod_j P(d_i|h_i)\right) \times P(h_i) = \sum_i \left(\prod_j \frac{\omega(d_j)}{\omega(h_i)}\right) \times \omega(h_i). \quad (2.3)$$

Finally, the prior likelihood is the given probability of seeing the evidence given the hypothesis, and it can be calculated as $P(D|H) = \frac{|D \times H \in M|}{|D|}$, i.e. the number of times the evidence has been seen together with the hypothesis divided by the number that the evidence has been seen in general.

Having all of the components of the inference, we get the final formula:

$$P(H|D) = \frac{\frac{|\omega(D\cap H)|}{|\omega(D)|}\times \prod_i \omega(h_i)}{\sum_i(\prod_j \omega(d_j)/\omega(h_i))\times\omega(h_i)}. \quad (2.4)$$

Using this formula, we can calculate a probability for an event to occur with the attributes in the H given some conditional parameters D. The probability for an event to occur given the instance data is defined as $P_a(a_{occured=1}|V_{e_i})$, where a is the node of the event for anomaly evaluation and $V_{e_i} = \{x: \exists e_i[e_i \in E \wedge \mu(e_j) \in V_e \wedge a \succ e_i \wedge x = e_j]\}$. Prediction of BP execution with a belief network can be done by finding the most probable next event given the current instance data, i.e. $argmaxP_p(a|V_{e_i})$, where $V_{e_i} = \{x: \exists e_i[e_i \in E \wedge \mu(e_j) \in V_e \wedge x = e_j]\}$ is the data of the current execution and $a: a \in E$ is an unknown event.

2.4. Simulation Model Generation from Bayes Belief Network

In order to simulate BPs, a simulation model first needs to be created. Standard simulation models are static, i.e. all elements and the control flow of the process is known before-hand. The static models use distribution probabilities for control flow or statistical variations in the duration of activities to simulate the pathways during the execution. The event logs contain more data than that – they contain a set of data attributes that describe each event in the log. These attributes, in real-life, are used in making decisions, or they might govern how long an activity takes time to complete. In general, the data describes the behaviour of BP. For this reason, the simulation model must also be capable of reflecting not only the control flow but also the data created during the execution of BPs. Such model with data taken into account facilitates making deeper insights on how the process behaves. The DBPS (see section 1.3.1) simulates the processes without a predefined control flow, activity execution activiation is based on conditions.

2.4.1. Dynamic Business Process Simulation

Dynamic BPs do not have a pre-defined sequence of steps (Kalibatiene *et al*., 2016). In declarative or imperative process modelling languages, process execution is managed using control-flow, therefore limits the behaviour (Fahland *et al*., 2010). As opposed to this, in dynamic BP execution activities are activated based on rules applied to the context, instead of a control flow mechanism, and it adds further flexibility to the BP execution or permits more complex decisions based on the data generated during the execution of BPs.

Due to its ability to use data generated during the execution of the processes and ability to use complex contextual rules for controlling the execution of the process (Vasilecas, Kalibatiene, *et al*., 2016), the DBPS was chosen as the simulation approach of the method. The simulation process is depicted in Fig. 2.7.

Definition 13. *The DBPS model is defined as a tuple* $S = (p, A, T, C, Q)$*:*

- *Context* C *is a set of elements* $c \in C$*, where* $c = (k, v)$ *with* k *- name of context element and* v *- element value. Context is used to model resources, simulation statistics and other parameters.*
- *Process* p *is a set of Activities.* $p = (a \in A)$*, where* A *is a set of possible activities defined in the process model or dynamically added/removed during simulation.*
- *Rule* r *is a function whose result states whether all conditions to execute an activity are satisfied.*
- *Activity* $a = (r, T_a \in T)$ *is a tuple, where* T_a *is a set of tasks which have to be executed if rule condition* r *is satisfied and* T *is a set of all possible tasks in simulation model.*
- *Task* $t \in T_a$ *is a function which can change any element of simulation state, e.g. activities, resources or context. Task is an atomic simulation action that can be neither divided nor interrupted.*
- *Queue* Q *is a set of elements* $q \in Q$*, where* $q = (t, d)$*.* t *is the task to be executed and* d *is the moment in time when the task should be executed.*

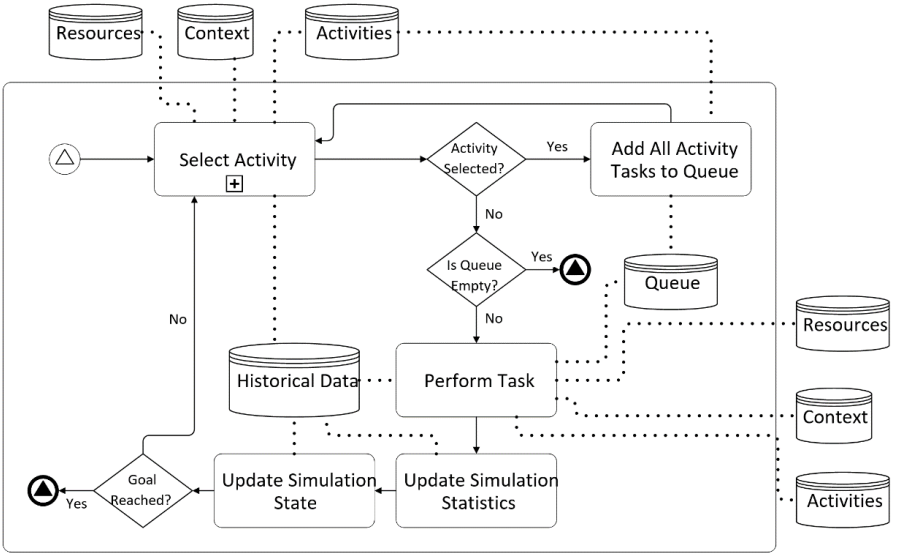


Fig. 2.7. A process on how simulation of dynamic business process is performed (Vasilecas *et al*., 2015)

When the simulation engine evaluates all information (*Resources*, *Context*, *Activities* and *Historical Data*), it selects the most suitable activity to be performed next. Once selected, tasks defined in the selected activity are added to the queue. Afterwards, activity selection subprocess repeats. If additional activity is found and selected, it means that these two activities are to be performed in parallel. The steps continue until no other activity is selected to be activated. When there are no more activities to start, a task queue is evaluated. If the task queue is empty, the simulation engine waits for a context change and resumes simulation once the change is detected. If the task Queue is not empty, then task(s), simultaneous in the next simulation step, are executed. During the step *update simulation statistic*, metrics such as duration, queue information, context variables are stored for further analysis. The activity *Update simulation state* updates variables related to the simulation execution: simulation time and task queue. After the state of simulation has been changed, a check is performed to see if the simulation goal is reached. If the goal has not been reached, the activity *perform simulation step* is started from the beginning with new or updated simulation variables. Otherwise, if the goal is reached, the simulation is completed.

2.4.2. Bayes Belief Network Transformation to Simulation Model

The conditions in DBPS models are defined in a predicate logic. Therefore, they allow data to be used in the activity selection decisions. Also, the DBPS allows activities to change the context, which itself is a set of data attributes. Such data-based behaviour fits neatly with the BBN behaviour – the BBN operates with probabilities of data, where the data is the causal reason for something to occur or not. These are the reasons that make the DBPS the choice of simulation for the proposed method.

The method to generate simulation model from an event log L is based on the following sequence of steps:

1. A belief network is discovered from an event log (section 2.3).
2. Using the belief network, the simulation model S_L is generated.
3. A belief network is combined with transition system to facilitate inferences on the events to occur during the simulation.
4. The simulation model is manually customised for specific needs. This step requires human input.

The method performs multiple transformations and objects of each step are associated with the objects in the following steps (Fig 2.8). Everything starts with a *log*. A *log* is a set of *traces* where each *trace* describes how an instance of a *process* has been executed. A *trace* contains one or more named *events*, and each

of them can have *data attributes* that are specific to that event. The following attributes are standard *data attributes* that apply to all events in all event logs:

- *Concept:name* of data type *string* defines a name of an event.
- *Time:timestamp* of data type *date* defines when an event has occurred.

Other data attributes are domain- and event-specific. The domain-specific data attributes, when used in belief networks, allow a probabilistic view of event occurrences and the data attributes they have.

Belief Network is a *Directed Acyclic Graph* (DAG) where each node has a *Conditional Probability Table* (CPT). When a belief network is discovered from a *log*, each node in a DAG of the belief network is named after corresponding *event's concept:name* data attribute. Each *node* in the belief network can have one or more relations that define the conditional dependency to other *nodes*. Each *node* has a *CPT* that is constructed from all occurrences of corresponding *events*, their associated *data attributes*, and preceding *events* with their *data attributes* in the same *trace*. The *CPT* is a table, therefore it consists of *Cells* where each *cell* defines a *data attribute's value*, and each *Row* has a *probability* to occur.

First of all, for the creation of the simulation model, the associations of elements of the belief network, simulation model and event log are defined:

- $evg: P \to M, \quad evg(p) = \{(a,b): a \in N \wedge v \in V \wedge n = concept:name\}$ is an injective surjective function mapping each node p of the belief network graph G to a specific name of events in the log;
- $aca: A \to P$ is a function mapping each activity a of simulation model s to node p of a belief network graph G;
- $gen: A \times C \to M \; where \; gen(a,c) = x \in \{m: e \in E, m \in \mu(e) \wedge evg(aca(a)) = name(e)\}$ is a function generating random attributes with values from previously observed attribute list of the events represented by the activity a;

When a simulation model is being created, a belief network's *DAG* is a direct source of the initial *Business Process (BP)*. Each *node* in the *DAG* is transformed to *Business Process Element (BPElement*). As an *event* in a *log* can represent any *element type (BPElementType)* of a process, its type is, by default, selected to be *Activity*. If it is not the correct *element type*, the user of the simulation could set it to another appropriate type. This way, the initial set of activities is created:

Definition 14. *A_{init} is the initial set of activities, where* $\forall a \in A_{init}: \exists aca(a)$

Each activity in the simulation model consists of a set of tasks and a rule that defines when an activity can occur. The rules are in the form of a predicate logic and define a condition, which, when satisfied, allows for an activity to occur. The

idea of the method is to automatically generate a simplified simulation model which could further be customised. Therefore, the simulation model focuses on the BP behaviour such that during the simulation, activities occur when in reality they would have the highest probability of occurrence. That is why, when a node is transformed to an activity, the rule of the activity to occur is defined as:

Definition 15. *A default activity rule r is defined as* $("started","true") \in C \wedge evg(aca(a)) = name(arg \max_{e \in E \subseteq L} P(e|C))$.

In other words, if the simulation is started and, based on the current state of the process, the most probable next event is the event represented by the activity, then this specific activity should be executed next.

The body of the activity is a task that invokes a belief network and randomly generates data attributes based on the CPT in the belief network for that specific node. The data generation function can be of any type – inferred most probable value set, weighted random selection, or other. The generation function should be selected based on the task the simulation tries to solve.

If the simulation tries to imitate intricately a specific process instance, then the generation function should use the inferred most probable value set. This would allow to see how changes of a specific process context would impact the behaviour of the process instance. On the other hand, it would limit the scope of the behaviour exposed by the simulation, and the same input variables would result in the same behaviour of the process instance. This limits the use of the generation function to a specific scenario testing.

The method attempts to expose as much behaviour as visible in the event log during the simulation to facilitate a Monte Carlo simulation, therefore weighted random selection generation function is the most appropriate. When data is generated during the simulation, all possible values sets of a specific node are selected and their probabilities, based on the current state of the process, are calculated. Afterwards, the random value set is selected weighted by the probabilities. It makes it more likely to select values for the most probable value sets and less likely for the rarer value set.

It is for the above reason that the method uses random weighted selection to generate data attributes. The resulting data attributes from the generation are then added to the context of the process. The exception to this is if one of the data attributes is named *duration*, in which case it is used as the duration of the task. Otherwise, the activity has no duration, i.e. it is instantaneous. The default task is defined as follows:

Definition 16. *A default task of an activity is such operation, which modifies context in such a way* $C' = C \cup gen(a, C) \backslash ((duration, x \in V) \in M)$.

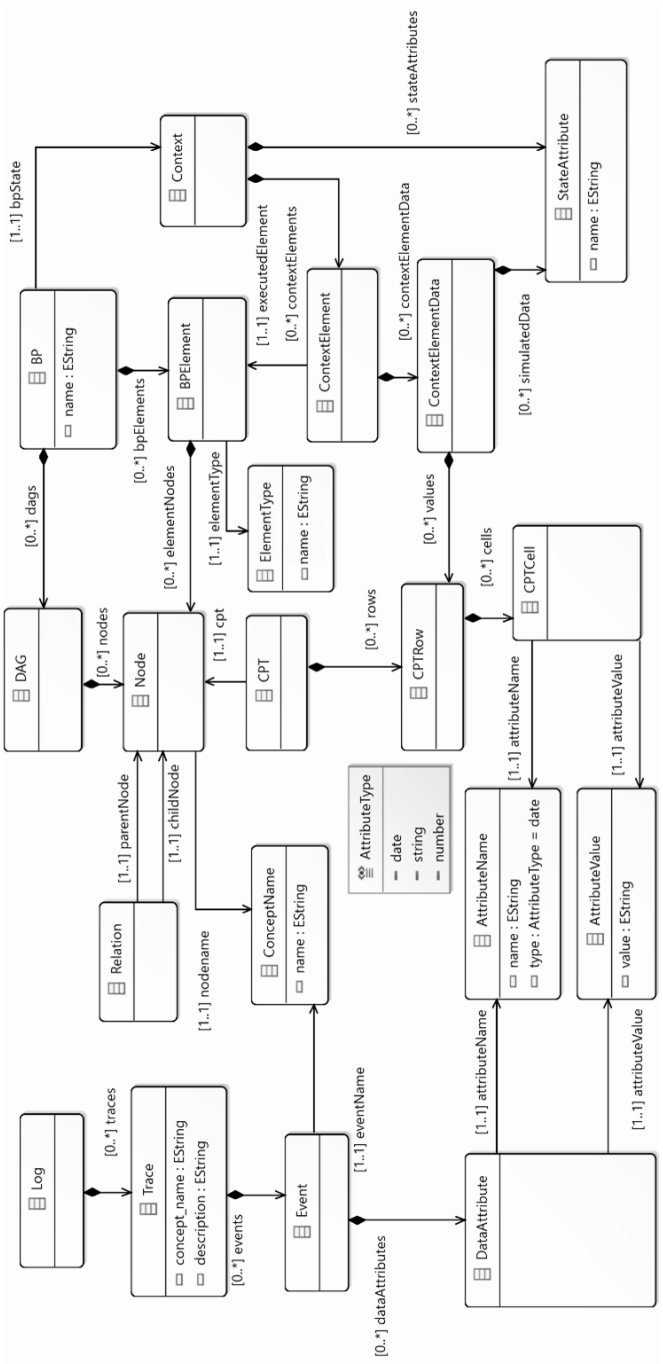


Fig. 2.8. Relations between the elements of an event log, belief network and simulation model in the proposed approach

The simulation model consists of multiple activities that, once executed, modify the context of the process. The context of the process describes the current state of the process instance. There are many possible types of context variable types, sources and values which are usually domain specific. For example, a context variable might be a result of a query to a server, a data record from the database, a file or anything else. In general DBPS use, the context is a set of variables that identify the activities that have been executed, the data that has been generated during the execution of the activities, the variables used by the activities, the events that have occurred, etc.

For the purpose of the method, it is assumed that the context is a set of key-value pairs with the name of an element and a value to synchronise it with the data used by the BBN.

Finally, once generated, the DBPS model can be customised based on the needs of the simulation performers. The customisations can be an addition of other activities, modification of activation rules, and/or modification of activity behaviour. For example, if there is a need to customize the model and add a rule such that a warehouse accepts incoming transport only after *9:00am* and only until *6:00pm*, an activity named *accept transport* could be modified to make its activation rule as (in pseudo code) *"$time > (9:00am) and $time < (6:00pm) and $started =true and most_probable_next_event(accept transport) = true*" .

Another example could be that the performers want to test only a single decision path. In that case, generated activities could be modified to generate a specific set of data attributes, thereby forcing a specific process path, as opposed to the usually used pseudo-random data. This way, the method allows for the performers of the simulation to automatically generate the initial simulation model from an event log and then modify it for their needs to test what-if scenarios or analyse the general process behaviour.

2.5. Conclusions of Chapter 2

1. The deliberation on the selection of a method for prediction and simulation has shown that Bayesian Belief Networks are useful for modelling stochastic domains, and in the case of the proposed method, stochastic properties of BPs and the underlying Directed Acyclic Graph in the proposed method can represent event dependency.
2. Automated BP simulation model generation allows reducing the workload required to prepare simulation models. It does not create a self-sufficient simulation model for all analysis tasks but facilitates further customisations to solve analytical tasks using BPS.

3. The presented Bayes Belief Network discovery method and inference formulas facilitate BP analysis in the form of decision support by inferring the probability of the current process instance state and prediction of BP execution to identify how the BP instance will behave in the short-term.
4. The main pitfall of the presented method is the limited BP behaviour exhibited during the simulation, because it focuses on data, but not on complex BP concepts, such as resource allocation rules.

3

Experimental Evaluation of the Proposed Method for Business Process Prediction and Simulation

The proposed method in the 2nd chapter is designed to support the BP analysis. There is a need to perform an experimental evaluation to prove that the method is usable in practice. This section presents the design of the experiment and the experimental results proving the usability of the proposed method. The contents of this chapter are based on previously published content by the author in (Kalibatiene *et al*., 2016; Savickas & Vasilecas, 2014, 2015, 2017; Vasilecas, Kalibatiene, *et al*., 2014; Vasilecas *et al*., 2015)

3.1. Experiment Design

There are multiple activities involved in creating the BBN and the Simulation model. Therefore each of the activity results must be verified. For this, the experiment has been designed to test each of the components of the activities in the method.

3.1.1. Selection of Input for the Experiments

The design of the experiments starts with the definition of the experimental data. The approach has a specific goal to be usable with event logs and to facilitate faster analysis. Therefore the first step in design is to select the input of the experiments – the event logs. The experiments should cover the method as much as possible, and the experiments must be verifiable. Therefore the following requirements apply to the event logs used in the experiments:

1. One or more of the event logs must be publicly available to facilitate repeatability.
2. One or more of the event logs must cover real-life use cases, i.e. they must belong to real-life processes.
3. There must be at least three event logs with differing process complexity to cover differing behaviour of the BPs.
4. The logs used in the steps of the experiments must overlap, i.e. experiments steps should re-use the same logs in each step.

Multiple event logs were selected to cover the requirements. First of all, a publicly available insurance claim synthetic BP event log (Van Der Aalst *et al*., 2007) was used. This log contains a total of 3512 cases with up to 11 events each. Not all events in the log contain data attributes, therefore some additional attributes, such as performer and age, were generated to imitate domain data. Secondly, 3 publicly available real-life logs from BPI challenge were used – a publicly available Dutch Financial Institution event log from BPI‘12 (Van Dongen, 2012), BPI’13 event log (Steeman, 2013) of car manufacturer incident reporting process and BPI ’15 log (Van Dongen, 2015) of municipality building permit application processes and. Finally, a proprietary university edict event log (EIMSD) of a fully dynamic process was selected, i.e. the sequences of the BP are not pre-defined. The logs contain 3512–1087 BP intances and 20339–262200 events with 9–289 possible vents and 2–12 attributes. The exact properties of the event logs are listed in Table 3.1.

Table 3.1. Properties of the event logs used in the experiments

Log	Instances	Unique Events	Total Events	Attributes
Synthetic log	3512	9	20339	2–6
BPI 12	13087	36	262200	3–4
BPI 13	7554	13	65535	9
BPI 15	1156	289	59083	12
EIMSD	2774	63	21392	6

3.1.2. Experiment Environment

To facilitate the experiments, an implementation of the proposed method is necessary. For this purpose, a prototype tool called BBNGs (Business process Belief Network enGine) was designed and implemented using .NET framework for creating BBNs and making inferences. The BBNGs is a tool designed to receive an event log of a BP as an input and transform it into belief network, thereby allowing inferences to be made using the belief network. The overall architecture is presented in Fig. 3.1.

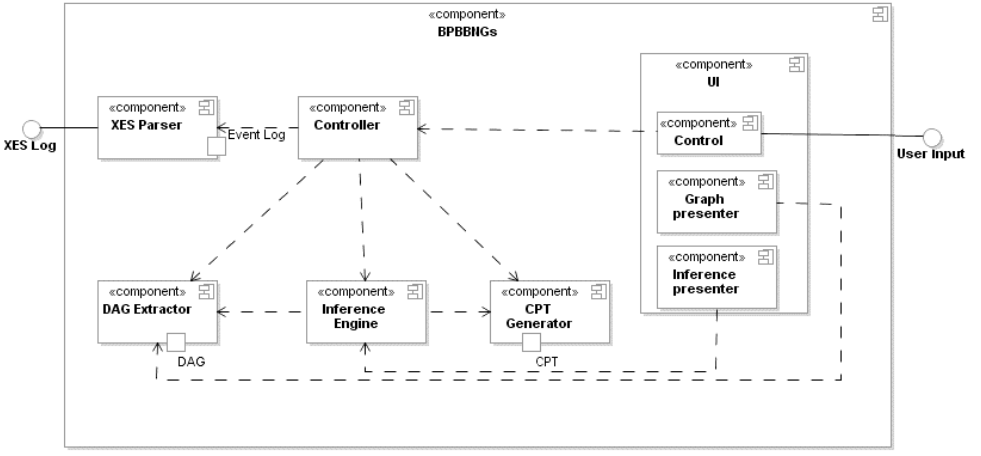


Fig. 3.1. Implementation architecture of the prototype

The main component responsible for the behaviour of the tool is the *controller*. This component is responsible for controlling the process of BBN discovery and initiating other actions, such as inferences or observations. It is connected to the *XES parser* component, which is responsible for loading event log files and pre-processing them to prepare for the discovery of the BBN. The steps of the proposed method are implemented in the *DAG Extractor*, *CPT Generator* and *Inference Engine* components. *DAG Extractor* component discovers directed acyclic graph from the event log, and then the *CPT Generator* component uses the event log to create conditional probability tables and connects the tables to the DAG. Finally, *Inference Engine* uses the DAG and the CPTs to make inferences and stores current variable observations.

To facilitate the use of the prototype to general users, the Prototype exposes the *User Interface* (GUI) (Fig. 3.2). The GUI is used by the users to perform actions such as:

– to load the event log source files,

- to perform attribute value observations, thereby creating the state l^{state} and l^{event},
- to execute inference or to test the correctness of the predictions,
- to preview discovered graphs and inference results.

The GUI has three main components – *Control*, *Graph Presenter* and *Inference Presenter*. The *Control* component is responsible for exposing functions related to the loading of source files, initiating BBN discovery, performing inferences and processing the discovered elements. The *Graph Presenter* component is used to provide a graphical view of the discovered directed acyclic graph, and the *Inference Presenter* provides a graphical representation of the inference results.

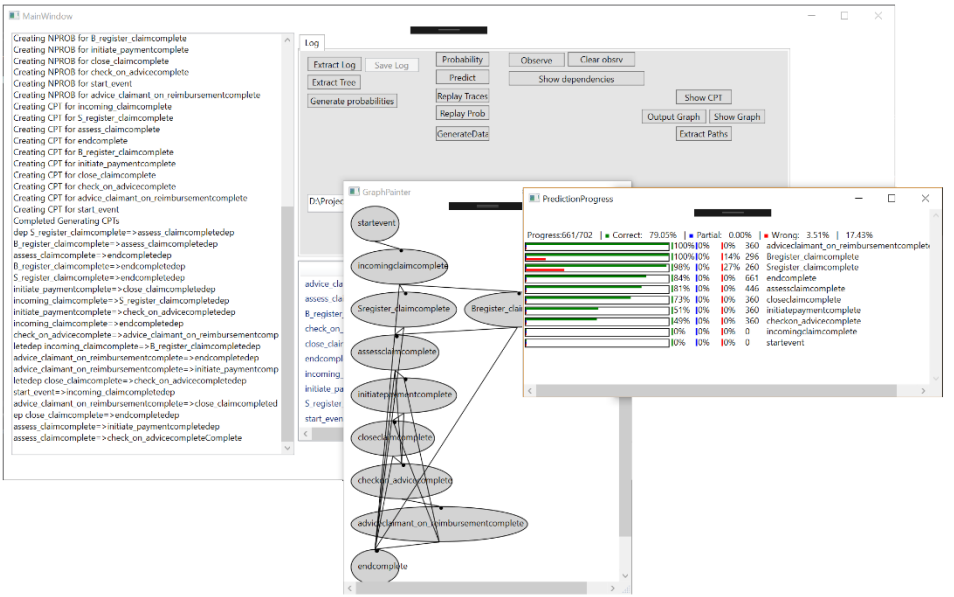


Fig. 3.2. The graphical user interface of the prototype

The initial task of the prototype is to receive an input in the XES format from an external system. The specific input format was chosen because most of the time ISs do not have event logs of BP and the data might be heterogeneous, and the XES file format is the most widely used format for representing the event logs. The *XES parser* component loads the data into the BBNGs and makes the event log accessible in memory. Afterwards, the other components perform each of the activities described in section 2.3.

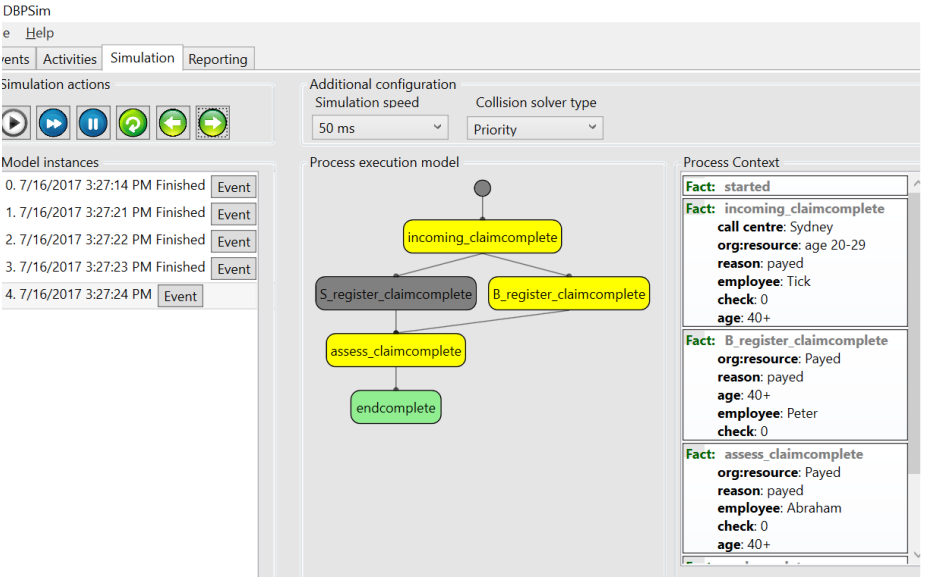


Fig. 3.3. DBPSim Simulation using integrated BBNGs module

The prototype BBNGs is used for working with BBN and has no simulation functionality. For the simulation, the BBNGs has been integrated into DBPSim (Fig. 3.3) – a prototype tool for DBP. The tool has been conceptualised and implemented in Vilnius Gediminas Technical University IS Research Laboratory. During the integration, DBPSim was extended with functionality to allow loading BBNGs as a plugin and to create a simulation model as defined in chapter 2.

3.1.3. Experimental Evaluation Steps

The final step in the experiment design is the definition of experiment steps. First of all, the approach discovers a BBN from an event log. This discovery consists of two steps – a DAG and CPT creation. The purpose of the DAG is to represent a BP albeit without cycles, therefore the first step is to verify the creation of DAG from event logs. Since the DAG is only a visualisation of event sequences, it is deemed sufficient to inspect the graphs visually and compare with alternative methods. Heuristic Miner (Weijters *et al*., 2006) was selected as a comparison because it is one of the most widely used methods to discover BP models.

The CPTs are used in inferences, therefore, to evaluate them, the next step is the experiments with the created BBNs to verify inference correctness. The evaluation is done with the different event logs using the following process:

1. The event log is transformed into two subsets – 90% and 10%.

2. The 90% subset is used for discovering BBN.
3. A leftover subset of the remaining 10% traces is used for the evaluation.
4. The experiment is repeated 9 more times with a different subset of event logs to create *k-fold=9* results.

The sample size of 90% for the training set is selected in order to cover the most behaviour available in the event log and to facilitate high k-fold validation. By choosing a smaller subset for training, the experiments could have worse results, but this would not test the method itself – the results would be worse only due to the fact that the BBNs express less behaviour than is available in the logs.

The experiment itself is performed by imitating the execution of a BP. The system iterates through each event and creates a partial trace with a state $l^{state}(\sigma)$, where σ is the currently iterated part of the trace in the event log. Already-knowing what is the state $l^{event}(\sigma(k))$ of the last event, $P\left(l^{event}(\sigma(k))|l^{state}\left(hd^{k-1}(\sigma)\right)\right)$ – the probability for the event's state, given the already occurred events in the partial trace – is calculated. The probability calculations are done only when $|\sigma| > 0$, i.e. at least one event is available in the partial trace. It is done so, because the first event in the trace has no impact in the experiment – its probability does not have any conditional dependencies, therefore it does not test the method.

For the evaluation of the method, we perform experiments which answer the following:

1. What are the average probability and standard deviation for the currently executed event?
2. How successful is the approach in predicting near-term future:
   a. with different size of attribute sets? In other words – what is the impact of the data on the effectiveness of prediction?
   b. with different log complexity? In other words – how does the method deal with different processes?

The evaluation should examine the following prediction results:

- Correct prediction, when the event was chosen correctly, i.e. it was correctly predicted that it would be the next one in the process instance.
- Correct interval prediction, when the chosen event was not the next one, but it occurred further in the process instance. Only a single correct interval prediction is counted to account for a possibility that the prediction is repeated multiple times until the event occurs.
- Wrong prediction, when the chosen event never appeared in the BP instance while it was predicted to occur.

The final step is the evaluation of the BP simulation. It was decided to apply the approach as depicted in Fig. 3.4 to evaluate whether the method is suitable for simulation and whether its results are satisfactory. The approach consists of the following steps:

1. Load an event log.
2. Generate simulation model based on the event log.
3. Perform simulation for at least 250 cases and generate event log of the simulation results (GEL).
4. Apply conformance checking methods to test source event log (SEL) against the simulated event log:
   a. discover Petri Net process models from the generated (GMM) and source event logs (SMM) using Inductive miner (Leemans *et al*., 2013).
   b. check conformance of the event logs versus the discovered Petri Nets using replay (Van Der Aalst, Adriansyah, & Van Dongen, 2012) for:
      i. the source event log against Petri Net discovered from the generated event log.
      ii. the generated event log against Petri Net discovered from the source event log.
5. Evaluate the results.

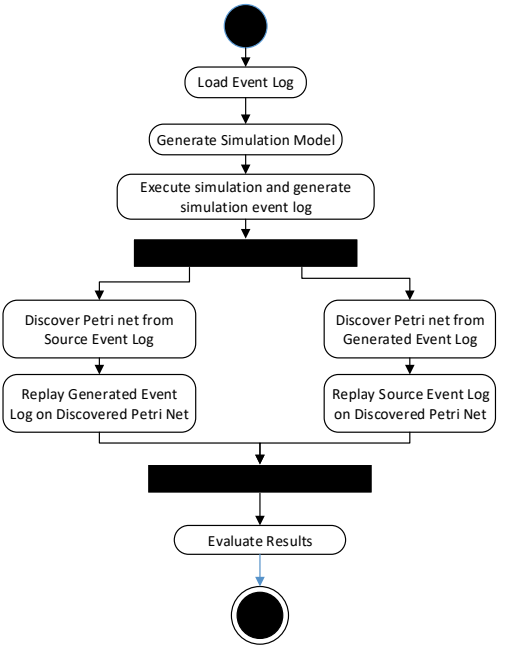


Fig. 3.4. Selected simulation evaluation approach

The chosen evaluation approach facilitates formal evaluation of the simulation results. Inductive miner was chosen for the discovery of Petri Nets because it is guaranteed to provide a Petri Net which is sound and fits the event log (Leemans *et al*., 2014). For conformance checking, it was decided to replay event logs against the discovered Petri Nets (Adriansyah *et al*., 2011b) to see whether the simulated results are conformant to the original sequences of events in the source event log.

The replay does not take into account generated data, but this does not need to be verified since the data does not strictly follow causal path – it is pseudo-randomly generated, i.e. it is generated not based on what is the most probable data, but weighted against probability distribution in the CPT.

3.2. Experimental Results

3.2.1. Evaluation of Bayesian Belief Network

During the processing of the event logs, attributes *lifecycle:transition*, *concept:name*, *time:timestamp* were ignored in the CPT creation step. In addition, each event had an attribute *time* added with timestamp's hour of the day. This was done to eliminate the amount of unique time values and include a possibly valuable data attribute. All other data attributes available in the log and associated with the events were used.

3.2.1.1. Evaluation of Directed Acyclic Graph Discovery

To test the directed acyclic graph discovery, the event logs were fed to the prototype and directed acyclic graphs representing BPs were discovered. The graph of the synthetic log discovered using the proposed method is present in Fig. 3.5a

The extracted graphs were compared to those extracted by the well-known and widely used PRoM tool (Van Dongen, De Medeiros, Verbeek, Weijters, & Van Der Aalst, 2005) and the algorithms it contains (Van Der Aalst, De Medeiros, & Weijters, 2005b; Van Der Aalst *et al*., 2004; Van Dongen, De Medeiros, & Wen, 2009). A sample graph discovered from the synthetic log using Heuristic miner is presented in Fig. 3.5b. The resulting graphs, as can be seen from the visualisations, are similar. The main difference is that the proposed method discovered a graph which has no cycles, as opposed to that of the Heuristic Miner's result.

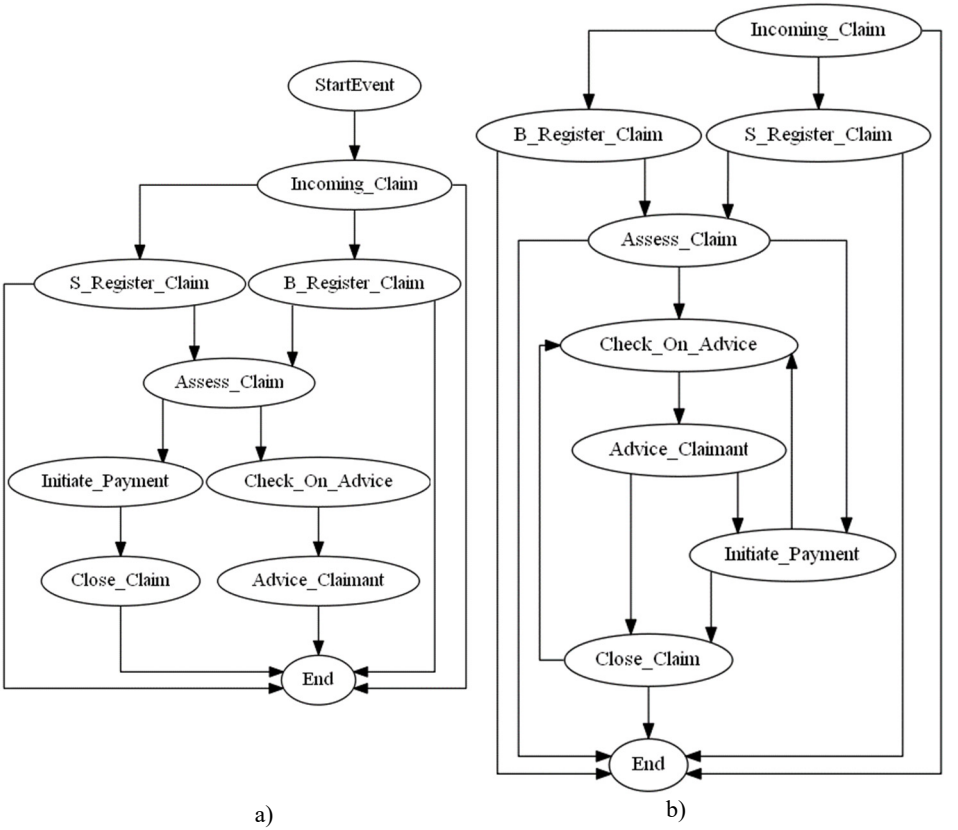


Fig. 3.5. Visualizations of discovered graphs of synthetic log: a) using the proposed method, b) using Heuristic Miner

3.2.1.2. Evaluation of Inference

The experiments of the inference resulted in a total of 17 750 trace runs with a total of 232 861 events. Not all of the events were used in probability calculations, and some of the probabilities were rejected. In total, 74 168 were rejected. The reason for the rejection was one of the following:

- the events had no data available in the training set;
- the events were anomalous ($P(D|H) = 0$);
- the events were the first event in the trace.

The precision defines the average expected probability of any event in the presented event log and is calculated as follows:

Definition 17. *given an event's probability function* $Prob(trace, eventPos) = P\left(l^{event}(\sigma(eventPos))|l^{state}(hd^k(\sigma) \in trace)\right)$, *average inferred probability (precision) is calculated as* $AV_{Log} = \frac{\sum_{k=0}^{k=|I|}\sum_{j=1}^{j=|e\in\beta(i_k\in I)|} Prob(i_k,j)}{\sum_{k=0}^{k=|I|}\sum_{j=1}^{j=|e\in\beta(i_k\in I)|} 1[Prob(i_k,j)=0]}$.

During the experiment, the precision was calculated for each of the event logs and with cross-validation of k-fold =10. The results are presented in Table 3.2.

Table 3.2. Probability inference results using *k-fold*=10

Log	Inferences taken into account	Total inferences/ Total events in the log	Events Observed/ Events in the log	Precision, %
Synthetic	10 679	16 782/20 339	8/9	78.71±22.73
BPI'12	122 720	147 255/262 200	34/36	63.1±14.26
BPI'15	11 236	53 324/59 083	35/289	98.16±16.36
EIDSM	14 058	15 500	58/63	62.66±10.25

From all of the inferences taken into account, the highest average probability was for the synthetic log, as expected. This was due to the fact that the underlying process is rather simple. The inference average was 78.71% with 4 of the events having an average inferred probability higher than 99% with deviation <1%. Other events had the average probabilities ranging from 30% to 85% with deviation ranging from ±36% to ±50%.

Other processes were much more complex and had much more possible data variations. This resulted in a lower average precision. In the case of the BPI'12 log, the events contained barely 3–4 data attributes, therefore the causal dependency between the events is arguable. This resulted in an average precision of 51%, but 10 out of 36 events had the average precision higher than 80%. 3 of the events in the BPI'12 log were ignored in inferences, because either they were always the first event or had occurred less than 5 times in each of the test sets.

The BPI'15 log had the lowest results regarding calculations taken into account (11 236 out of 53 324), but the process itself is the most complex – the log has 289 unique possible events and only 1 156 traces in total. This causes the belief network to perform under-trained due to such complex structure and low amount of data. Also, for the log, there were usable inferences only for 35 out of 289 possible events. Ignoring that, the average precision was 99%. Furthermore,

in total 33 out of 35 events taken into account had the average probability higher than 80%.

To sum up, the experiments show that the method is usable, but its effectiveness relies on data. When the values of an attribute are observed but have never been observed in the training set, or if there is a limited amount of historical observations which do not fully cover the process behaviour, the approach has limited use. In any case, for events whose behaviour is expressed in the event log, the proposed approach shows great results and allows answering questions important to the execution of processes – whether events are expected in the process instance or what data might accompany those events.

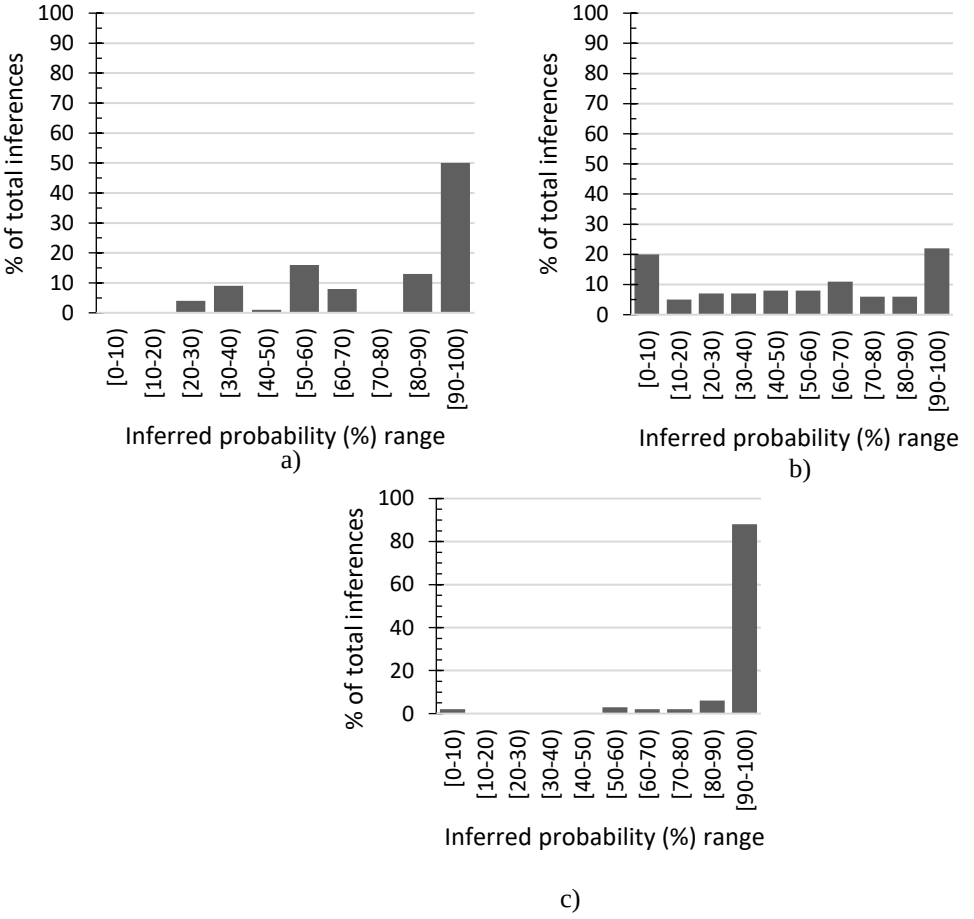


Fig. 3.6. Inference results of: a) Synthetic log; b) BPI'12; c) BPI'15

3.2.2. Evaluation of Prediction

The charts in Figure 3.7 show a list of events in the process log and their prediction effectiveness. On the left of the charts are bars whose width is relative to the value of the prediction result count. On the top are the total values for all predictions in the experimental set. The green colour represents Correct predictions, red colour – wrong predictions, and blue bar – correct interval predictions.

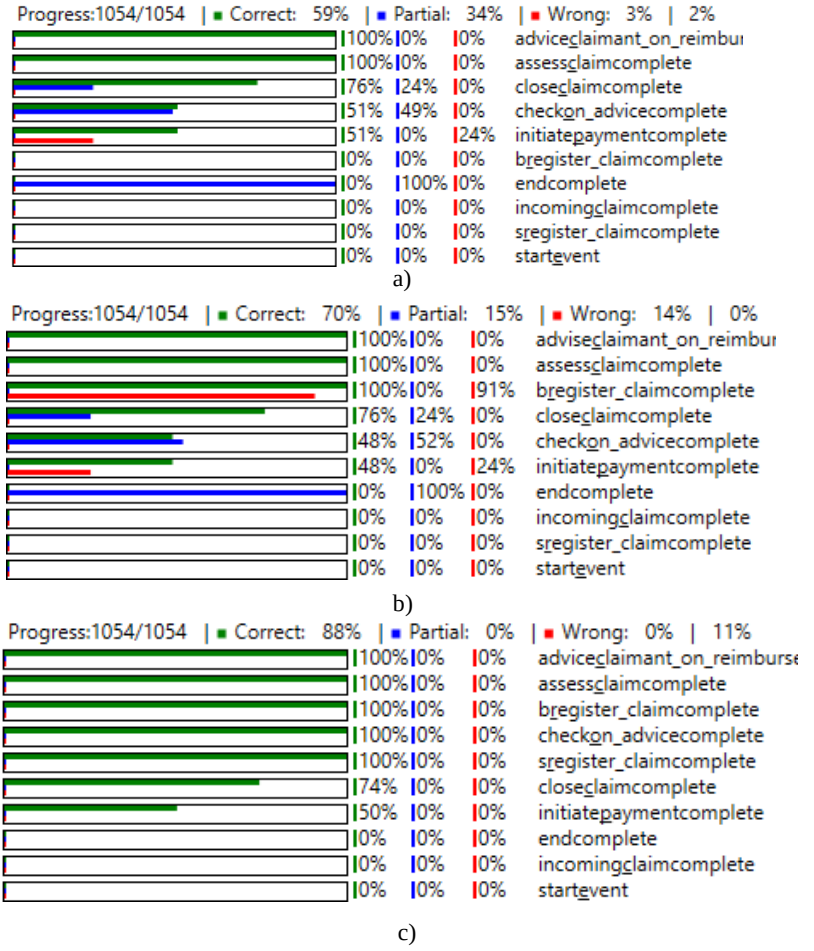


Fig. 3.7. Screenshots of prediction results using: a) weighted random selection; b) reduced attribute count; c) full attribute list

In the Fig. 3.7, there are always two events that contain no data. These events are the *startevent* and *incomingclaimcomplete*. These events are always at the first position in the traces. Therefore no data predictions are made for them.

The results of the experiment show that the proposed method is an improvement over random selection based on frequency. It is an improvement because random selection ignores the actual data that occurs in real-life processes. This data is ignored by most BP prediction approaches where only behavioural statistics are used for the prediction. It can be seen that the method is dependable on the attributes of each event. Therefore the correct selection of data related to the BP is crucial to the analysis results.

The next experiment step is the evaluation of the prediction capabilities. The results of the prediction experiment are presented in Figure 3.8 and Table 3.3.

Table 3.3. Prediction results for event-logs with k-fold of 10

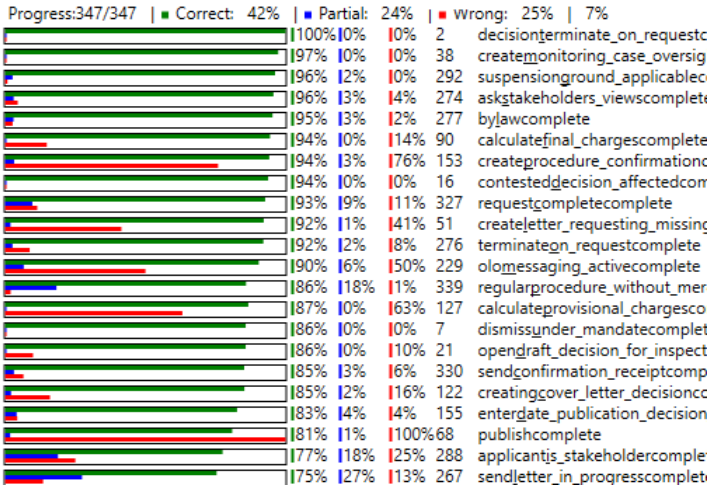
Event Log	Correct, %	Correct Interval Prediction, %	Wrong, %	Missing, %
Synthetic	84.49±18.66	2.52±7.29	5.74±8.06	7.25±19.67
BPI'12	73.49±31.36	3.61±11.31	13.7±18.86	9.2±25.04
BPI'15	53.01±31.54	21.36±23.27	18.24±22.55	7.17±25.01
EIMSD	67.6±32.54	4.87±10.66	21.99±31.57	5.54±30.47

The BPI'15 log prediction rate is the lowest. It managed to predict correctly on average only 53.01% of the events and 21.36% of events in incorrect sequence; also, on average it failed to predict 7.17% of events and made wrong prediction for 18.24±22.55% of the predictions. The low prediction rate is due to the high amount of unique event types – overall 398 – and a small training set. On top of this, the number of dimensions – count of event types and attributes – in comparison with total event count is insufficient for reliable predictions. There are too many possible value combinations with the regards to trace count. In other words, from BP perspective, the underlying process is complex.

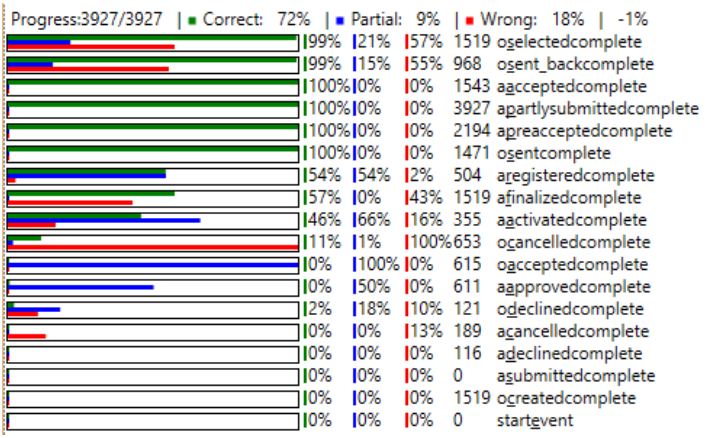
The prediction for BPI'12 log shows much better results. The approach, on average, correctly predicted 73.49% percent of events and 3.61% of events in incorrect sequence. It missed, on average, 9.2% of events and made wrong predictions in 13.7% of cases.

Finally, EIMSD was the most dynamic BP as the control flow was not fully controlled. In real life a decree for a student cannot always be decided, e.g. when a student leaves university, there is a decree for leaving, but the time and purpose of the leaving might be unknown from the data in the event log. It can be seen from the results that EIMSD log prediction was correct, on average, for 67.6% of events and wrong predictions were made for 21.99% events. The lacking results

might be partially explained by the underlying dynamic control flow of the process. Additional causally related attributes are required to improve the results.



a)



b)

Fig. 3.8. Screenshot of fragments of: a) BPI'15 log prediction;
b) the prediction of BPI'12 log

It can also be seen from Figure 3.8 that no matter what event log is used for the evaluation, the approach missed 5.54–9.2% of events. There is a need for deeper research on what are the root causes of the stable Missing and what could be done to improve the prediction.

3.2.3. Evaluation of Simulation

The simulation was executed using input event logs, called Source Event Log (SEL). The result of the simulation was a generated event log (GEL) which was used for further evaluation. The properties of the SEL and GEL are listed in Table 3.4.

Table 3.4. Parameters of the logs used in evaluation of simulation

Log	Traces	Unique Events	Total Events	Attributes
Synthetic log – source	3512	9	20 339	2–6
Synthetic log – generated	530	9	4 240	2–6
BPI12 – source	13087	36	262 200	3–4
BPI12– generated	530	34	4 526	3–4
BPI13 – source	7554	13	65 535	9
BPI13 – generated	524	10	1 231	9
BPI15 – source	1156	289	59 083	12
BPI15 – generated	250	136	38 364	12

As defined in the experiment design, Petri Nets were discovered from the event logs: Source event log Mined Model (SMM) was discovered from the SEL and Generated event log Mined Model (GMM) was discovered from the GEL. Exemplary SMM and GMM are presented in Figure 3.9. It is clear that the two discovered Petri Nets are very similar in their expressiveness of the control flow.

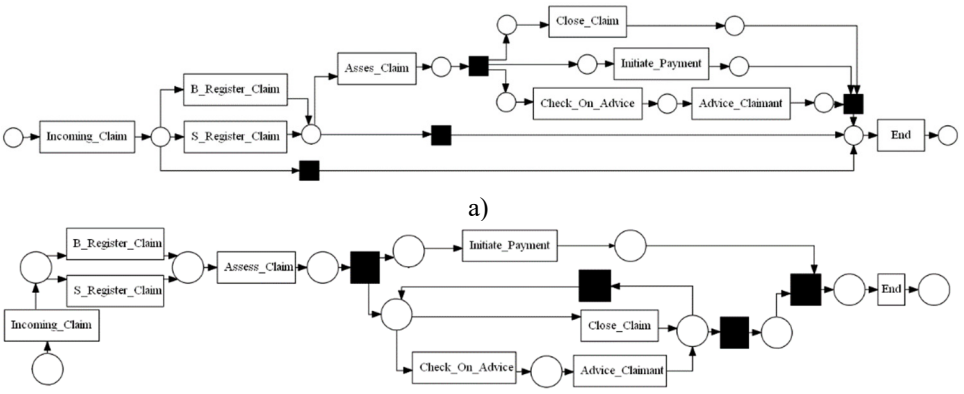


Fig. 3.9. Petri Net discovered: a) from Synthetic Log Source; b) from Generated Event Log; b) using Inductive Miner

The differences of the discovered Petri Nets are minimal, e.g. there is a missing path from *Incoming_Claim* to the *End* event in the GMM, while the arc is present in the SMM. All missing links are related to the jump to the *End* event from the middle of the process. Such jump indicates that the process was stopped unsuccessfully, and it does not happen often. Due to the low probability of the scenario to occur, it is possible that simulation was no run long enough to generate such cases.

Finally, conformance checking was done. It was done by cross-replaying the event logs on the two mined Petri nets to reduce any possible impact of the chosen discovery algorithm. As can be seen from the results in Table 3.5, SEL replay on GMM is less fitting, but this could be explained by the fact the simulation result had a lower amount of traces and because the log might have failed to cover all possible execution paths observed in the SEL. It is further proven by the fact that the simpler synthetic SEL had a high move-log fitness (0.929) on the GMM and high fitness (0.941 trace and 0.891 move-log fitness) of the GEL on SMM. In general case, the simulated processes had a fitness against the source event log higher than 0.814, except for the BPI13 event log, whose flow was dynamic and the data attributes did not represent decisions made in the process. Other than that, the other lowest fitness was for the BPI15 log, which has the biggest number of unique events and the least amount of traces, therefore it would be required to simulate many traces to cover all of the possible execution paths.

Table 3.5. Simulation evaluation results for different event logs

Log	SEL conformance with GMM		GEL conformance with SMM	
	Trace Fitness	Move-Log Fitness	Trace Fitness	Move-Log Fitness
Synthetic log	0.795	1.000	1.000	1.000
BPI12	0.821	0.817	0.814	0.771
BPI13	0.928	0.906	0.438	0.582
BPI15	0.572	0.585	0.886	0.832
EIMSD	0.982	0.995	0.979	0.987

SEL – Source event log, SMM – Mined Model from Source event log, GEL – Generated event log, GMM – Mined model from generated event log

As can be seen from the results, the fitness of the SEL against the GMM varies more widely than the other way around. This could be explained by the complexity of the process that is contained in the SEL and limited trace count in the GEL, e.g. there are 289 unique events in the BPI15 source log with 1159 traces while the generated log contained only 250 traces and 136 unique events. This means that the generated log did not express as much behaviour as the source log.

In order to evaluate whether the size of the event log has an impact on the expressiveness of the process and what is the least amount of traces to have in an event log to achieve stable fitness, different length of simulation was performed. The simulation resulted in differing amounts of traces. The tests were done with 100–1000 simulation runs for the BPI12 log. The results are shown in Table 3.6 and charted in Figure 3.10 (with trendlines).

Table 3.6. Simulation evaluation results for differing trace counts

Trace count	SEL conformance with GMM		GEL conformance with SMM	
	Trace Fitness	Move-Log Fitness	Trace Fitness	Move-Log Fitness
100	0.593	0.572	0.913	0.885
200	0.649	0.627	0.839	0.848
300	0.699	0.653	0.810	0.766
400	0.752	0.732	0.819	0.833
500	0.821	0.817	0.814	0.710
600	0.821	0.817	0.817	0.775
700	0.840	0.831	0.821	0.778
800	0.835	0.833	0.825	0.839
900	0.800	0.773	0.830	0.790
1000	0.830	0.88	0.820	0.778

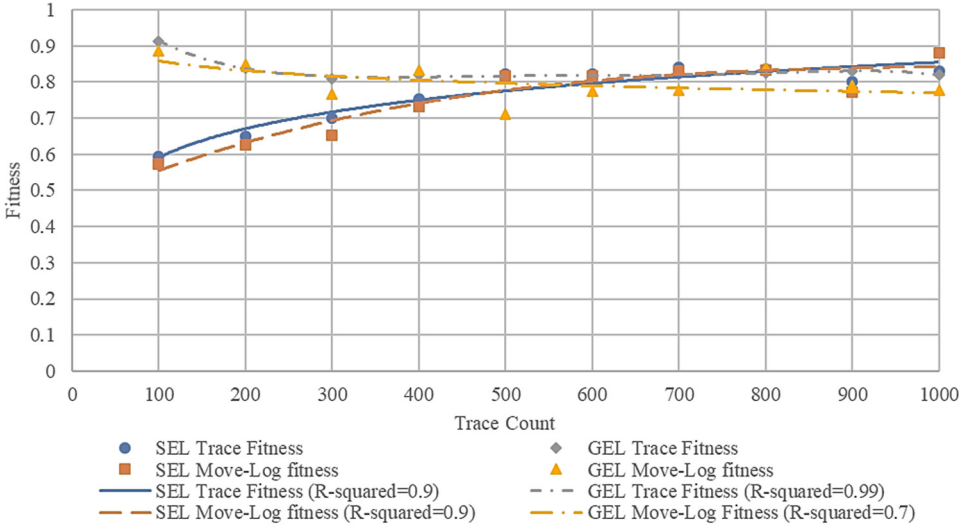


Fig. 3.10. Fitness versus trace count

The fitness of GEL against SMM log increases when the simulated trace count is decreasing, and the conformance of the source event log with Petri Net discovered from the generated event log decreases when the simulated trace count is decreasing. In both cases, the fitness stabilises at 500 traces. This can explain why the BPI'15 fitness results were so low – the process is very complex (has 289 unique events), and there were not enough simulated traces (250) to reach stable fitness results. This is further proven by the number of unique events that were in the generated event log – only 139 versus 289 in the source event log.

3.2.4. Threats to Validity

The experiments were done to evaluate the approach and its capability to solve the problems it is designed for. It was created based using constructive research method, therefore threats to the validity of the research must be addressed. The chosen validity threat classes were Construct, Internal, External and Reliability (Barros & Neto, 2011) and the assessment is presented in Tables 3.7–3.9.

Table 3.7. Construct Validity Threats

Threat	Management
Poorly chosen evaluation metrics	The metrics for evaluating the approach were chosen directly based on the problem the approach tries to solve. Performance and efficiency metrics were not selected because these metrics were not the focus of the research and are hard to control, i.e. they depend not only on the theoretical framework but also on the capabilities of the implementer and experiment environment.
Ineffective measures selected	For decision support, business analysts want to know how reliable their decisions are and for this, general inference correctness was evaluated. BP behaviour prediction can have multiple outcomes, therefore each of them was evaluated. The missing evaluation was on how correctly the data in the events are predicted, but this was not the focus of the current research. It was sufficient to know that CPTs contain the correct probabilities of each of the *(attribute, value)* set. Business Process Simulation deals with replicating real BP behaviour. The selected testing method used state of the art conformance checking technique. In order to eliminate possible errors of model discovery, a cross-check was done.
Bugs in implementation	The prototype was created by an experienced developer and the error count in implementation is minimal. The possible errors that could provide better than actual results are those that perform the evaluation, but the evaluation components are rather simple and were tested for errors during the implementation of the prototype, therefore bug risk is minimal.

Table 3.8. Internal Validity Threats

Threat	Management
Poor parameter settings	The experiment is described in detail and designed to maximise the coverage of the evaluation.
Lack of discussion on instrumentation	The prototype implementation is described.
Lack of clear of data collection tools and procedures	Data selection for experiments has been described in detail, and all parameters of the data are identified. The selected data was taken to cover as wide real-life application cases as possible, i.e. varying complexity event logs with differing number of traces, event types, attribute count and other properties.
Sensitivity to event log size	The approach is based on Bayesian Belief Networks, therefore calculations increase exponentially with increasing amount of data (event types, data attributes). For this reason, one of the logs in the experiment was sufficiently big to test this problem. It consisted of 289 distinct events. The approach was successful in inferring probabilities and simulating such process, therefore it is viable for real-life application.

Table 3.9. External Validity Threats

Threat	Management
Unreliable data	The data input of the experiments was selected to cover as wide range of possible scenarios as possible. The baseline scenario was covered by a synthetic log with a very simple process, thereby showing general applicability of the approach. Real-life data was taken from well-known public source – Business Intelligence Conference, which provides data sources that are used by researchers in the research area. Finally, one more non-public dataset (EIMSD) was chosen to cover a process which is unpredictable, i.e. does not have clear control flow.
Limited behaviour exposed in the event logs	The approach is sensitive to the data existing in the event logs. When the ration between exposed behaviour (such as event types) and trace count is high, the success of the approach is limited.
Non-comparable experiment	The approach and experiment are described in detail, and some of the data sources used are publicly available, therefore experiments can be done to compare the approach with other research.
Lack of evaluations for instances of growing size and complexity	The data selected for experiments was taken from publicly available sources and had a wide range of data in it (trace count-wise and data attribute-wise). Scalability evaluation was not the focus of the research, and it would be possible only with non–public data, thereby reducing experiment reproducibility.

3.3. Conclusions of Chapter 3

1. The BBN creation from an event log method allows decision support with an average inferred probability of events ranging 63–98% and prediction of events in the current business process instance with good (immediate or in the interval) predictions ranging 71–87%.
2. The proposed method of creating business process simulation models from a discovered BBN can simulate business processes with fitness ranging 58–98% for generated business process model versus source event log.
3. The method is reliant on information existing in event logs and complex event logs with many event classes versus a small number of traces or data attributes lead to unreliable results.
4. To achieve reliable simulation results, at least 500 simulation runs for a process should be run for simulation results to sufficiently exhibit the behaviour of the simulation process.

General Conclusions

1. The review of process mining techniques has shown that existing techniques are specialised, and this research area lacks general analysis methods. Also, the predictive process mining techniques focus on single parameters, such as duration and do not focus on domain-specific data attribute prediction.
2. The review of business process simulation methods and techniques has shown that although they provide value to business process analysis, they still have limitations – there is only a single BPS language standard, each tool has proprietary simulation models or requires specific skill-set. Dynamic Business Process Simulation (DBPS) attempts to facilitate flexible simulation with detailed models, therefore, could be integrated with knowledge discovered from event logs.
3. The proposed method for discovering Bayesian Belief Networks (BBN) from event logs automatically creates a probabilistic business process model, thus facilitating decision support via inference of the current process instance state probability and business process execution prediction.
4. The proposed method for transforming the BBN to DBPS reduces manual labour required for the creation of the initial simulation model from an

event log, and the simulation of the model reflects the source event log with high fitness.

5. The proposed approach of using Bayes Belief Network and Simulation Model Generation facilitates analysis of general BP behaviour, execution of currently running process instances, and the future of the process execution.
6. The experimental evaluation has shown that the proposed approach:
   a) is able to create DAGs;
   b) allows inference of the currently executed processes' current event's probability with the precision of 63–98%;
   c) allows prediction of events in the current business process instance with good (correct or partially correct) guesses ranging 71–87%;
   d) generates simulation models with replayability cost-based fitness ranging 58–98% for generated business process model versus source event log.

References

Ackermann, L., Schönig, S., & Jablonski, S. 2016. Inter-Paradigm Translation of Process Models using Simulation and Mining. *CoRR*.

Adriansyah, A., Van Dongen, B. F., & Van Der Aalst, W. M. P. 2011a. Conformance checking using cost-based fitness analysis. *Proceedings – IEEE International Enterprise Distributed Object Computing Workshop, EDOC*, 55–64.

Adriansyah, A., Van Dongen, B. F., & Van Der Aalst, W. M. P. 2011b. Conformance checking using cost-based fitness analysis. In *Proceedings – IEEE International Enterprise Distributed Object Computing Workshop, EDOC*, 55–64.

Aguilar-Savén, R. S. 2004. Business process modelling: Review and framework. *International Journal of Production Economics*, *90*(2), 129–149.

Ahn, S., Dunston, P. S., Kandil, A., & Martinez, J. C. 2015. Process mining technique for automated simulation model generation using activity log data. In *Congress on Computing in Civil Engineering, Proceedings*, Vol. 2015–Janua, 636–643.

Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Maggi, F. M., Marrella, A., Mecella, M., & Soo, A. 2017. Automated Discovery of Process Models from Event Logs: Review and Benchmark, 1–20.

Barros, M. D. O., & Neto, A. C. D. 2011. Threats To Validity In Search-Based Software Engineering Empirical Studies, 1–11.

Bezerra, F., & Wainer, J. 2013. Algorithms for anomaly detection of traces in logs of process aware information systems. *Information Systems*, *38*(1), 33–44.

Bizagi. (n.d.). Bizagi BPMN 2.0 – Process Modeling Standards. Retrieved from https://www.bizagi.com/uk/products/benefits/standards [Accessed: March 11, 2017]

Bose, R. P. J. C., Van Der Aalst, W. M. P., Zliobaite, I., & Pechenizkiy, M. 2014. Dealing with concept drifts in process mining. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(1), 154–171.

Business Process Simulation (BPSim). 2017. Retrieved from http://www.sparxsystems.com/resources/user-guides/simulation/business-process-simulation-bpsim.pdf [Accessed: July 17, 2017]

Ceci, M., Lanotte, P. F., Fumarola, F., Cavallo, D. Pietro, & Malerba, D. 2014. Completion Time and Next Activity Prediction of Processes Using Sequential Pattern Mining. *Discovery*, *8777*(1), 49–61.

Cherdantseva, Y., Hilton, J., & Rana, O. 2012. Business Process Model and Notation. *Lecture Notes in Business Information Processing*, *125*(March), 107–115.

Conforti, R., Dumas, M., García-Bañuelos, L., & La Rosa, M. 2016. BPMN Miner: Automated discovery of BPMN process models with hierarchical structure. *Information Systems*, *56*, 284–303.

Cook, J. E., & Wolf, A. L. 1998. Discovering models of software processes from event-based data. *ACM Transactions on Software Engineering and Methodology*, *7*(3), 215–249.

Darwiche, A. 2008. *Chapter 11 Bayesian Networks – Handbook of Knowledge Representation*. *Foundations of Artificial Intelligence*, Vol. 3. Elsevier.

De Leoni, M., Munoz-Gama, J., Carmona, J., & Van Der Aalst, W. M. P. 2014. Decomposing Alignment-Based Conformance Checking of Data-Aware Process Models. *On the Move to Meaningful Internet Systems: OTM 2014 Conferences SE - 1*, *8841*(i), 3–20.

De Leoni, M., & Van Der Aalst, W. M. P. 2013. Data-aware process mining. *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13*, 1454.

De Medeiros, A. K. A. 2006. Genetic process mining. *Cip-Data Library Technische*.

De Weerdt, J., De Backer, M., Vanthienen, J., & Baesens, B. 2012. A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. *Information Systems*, *37*(7), 654–676.

Di Ciccio, C., Maggi, F. M., & Mendling, J. 2016. Efficient discovery of Target-Branched Declare constraints. *Information Systems*, *56*, 258–283.

Eicher, J., & Ruder, D. 2007. Business process analytics: A new approach to risk. *Journal of Alternative Investments*, *10*(2), 76–84.

Erevelles, S., Fukawa, N., & Swayne, L. 2016. Big Data consumer analytics and the transformation of marketing. *Journal of Business Research*, *69*(2), 897–904.

Fadel, F. G., Fox, M. S., & Gruninger, M. 1994. A generic enterprise resource ontology. *Proceedings of 3rd IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, (April), 117–128.

Fahland, D., Mendling, J., Reijers, H. A., Weber, B., Weidlich, M., & Zugal, S. 2010. Declarative versus imperative process modeling languages: The issue of maintainability. *Lecture Notes in Business Information Processing*, *43 LNBIP*, 477–488.

Ferreira, D. R., & Alves, C. 2012. Discovering user communities in large event logs. *Lecture Notes in Business Information Processing*, *99 LNBIP*(PART 1), 123–134.

Ferreira, D. R., Zacarias, M., Malheiros, M., & Ferreira, P. 2007. Approaching Process Mining with Sequence Clustering: Experiments and Findings. *Lncs*, *4714*(1), 360–374.

Folino, F., Guarascio, M., & Pontieri, L. 2014. Mining predictive process models out of low-level multidimensional logs. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8484 LNCS*, 533–547.

Georgakopoulos, D., Hornick, M., & Sheth, A. 1995. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, *3*(2), 119–153.

Giuseppe, C., Valerio, M., Teresa, M., & Carmela, S. L. 2014. A Simulation Approach in Process Mining Conformance Analysis. The Introduction of a Brand New BPMN Element. *IERI Procedia*, *6*, 45–51.

Goedertier, S., Vanthienen, J., & Caron, F. 2015. Declarative business process modelling: principles and modelling languages. *Enterprise Information Systems*, *9*(2), 161–185.

Greasley, A. 2004. *Simulation Modelling for Business*. Routledge.

Griffeth, R. W., Hom, P. W., & Gaertner, S. 2000. A Meta-Analysis of Antecedents and Correlates of Employee Turnover: Update, Moderator Tests, and Research Implications for the Next Millennium. *Journal of Management*, *26*(3), 463–488.

Groves, P., Kayyali, B., Knott, D., & Van Kuiken, S. 2013. The "big data"revolution in healthcare. *McKinsey Quarterly*, (January), 22.

Guha, S., Kettinger, W. J., & Teng, J. T. C. 1993. Business Process Reengineering. *Information Systems Management*, *10*(3), 13–22.

Günther, C. W., & Van Der Aalst, W. M. P. 2007. Fuzzy mining–adaptive process simplification based on multi-perspective metrics. *Business Process Management*.

Hammer, M., & Champy, J. 2001. *Reengineering the Company – A Manifesto for Business Revolution*. *Harper business, New York USA*, Vol. 19.

Hammer, M., Vom Brocke, J., & Rosemann, M. 2010. Handbook on Business Process Management 1. *Handbook on Business Process Management*, 622.

Heidari, F., Loucopoulos, P., & Brazier, F. 2013. A meta-meta-model for seven business process modeling languages. *Business Informatics*, 216–221.

Hlupic, V., & Robinson, S. 1998. Business process modelling and analysis using discrete-event simulation. *1998 Winter Simulation Conference. Proceedings (Cat. No.98CH36274)*, *2*, 1363–1369.

Jansen-Vullers, M., & Netjes, M. 2006. Business Process Simulation – A Tool Survey. *Management*, *834*, 77–96.

Jensen, F. V. 1996. Bayesian networks basics. *AISB Quarterly*, 9–22.

Kalenkova, A. A., Van Der Aalst, W. M. P., Lomazova, I. A., & Rubin, V. A. 2017. Process mining using BPMN: relating event logs and process models. *Software and Systems Modeling*, *16*(4), 1019–1048.

Kellner, M. I., Madachy, R. J., & Raffo, D. M. 1999. Software process simulation modeling: Why? What? How? *Journal of Systems and Software*, *46*(2), 91–105.

Kim, G.-H., Trimi, S., & Chung, J.-H. 2014. Big-Data Applications in the Government Sector. *Association for Computing Machinery. Communications of the ACM*, *57*(3), 78.

Korherr, B. 2006. An Evaluation of Conceptual Business Process Modelling Languages. *SAC '06 Proceedings of the 2006 ACM Symposium on Applied Computing*, (Section 3), 1532–1539.

Laue, R., & Mueller, C. 2016. The Business Process Simulation Standard (BPSIM): Chances And Limits. *Ecms*, 413–418.

Leemans, S. J. J., Fahland, D., & Van Der Aalst, W. M. P. 2013. Discovering block-structured process models from event logs – a constructive approach. In *Application and Theory of Petri Nets and Concurrency*, 311–329.

Leemans, S. J. J., Fahland, D., & Van Der Aalst, W. M. P. 2014. Discovering block-structured process models from event logs containing infrequent behaviour. In *Lecture Notes in Business Information Processing*, Vol. 171, 66–78.

Liu, Y., Zhang, H., Li, C., & Jiao, R. J. 2012. Workflow simulation for operational decision support using event graph through process mining. *Decision Support Systems*, *52*(3), 685–697.

Ly, L. T., Rinderle-Ma, S., Dadam, P., & Reichert, M. 2006. Mining Staff Assignment Rules from Event-Based Data. *Business Process Management Workshops*, 177–190.

Maggi, F. M., Bose, R. P. J. C., & Van Der Aalst, W. M. P. 2012. Efficient Discovery of Understandable Declarative Process Models from Event Logs. *CAiSE*, *7328*, 270–285.

Maggi, F. M., Dumas, M., García-Bañuelos, L., & Montali, M. 2013a. Discovering data-aware declarative process models from event logs. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8094 LNCS*, 81–96.

Maggi, F. M., Dumas, M., García-Bañuelos, L., & Montali, M. 2013b. Discovering data-aware declarative process models from event logs. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8094 LNCS*, 81–96.

Martin, N., Depaire, B., & Caris, A. 2015. The use of process mining in a business process simulation context: Overview and challenges. *IEEE SSCI 2014 – 2014 IEEE Symposium*

Series on Computational Intelligence – CIDM 2014: 2014 IEEE Symposium on Computational Intelligence and Data Mining, Proceedings, 381–388.

Martin, N., Depaire, B., & Caris, A. 2016. The Use of Process Mining in Business Process Simulation Model Construction. *Business & Information Systems Engineering*, *58*(1), 73–87.

Morgan, C. B., Banks, J., & Carson, J. S. 1984. Discrete-Event System Simulation. *Technometrics*, *26*(2), 195.

Murphy, K. P., Weiss, Y., & Jordan, M. I. 1999. Loopy belief propagation for approximate inference: An empirical study. *Proceedings of Uncertainty in AI*, *9*(4), 467–475.

Nagatou, N., & Watanabe, T. 2015. A model-checking based approach to robustness analysis of procedures under human-made faults. In *International Journal of Industrial Engineering : Theory Applications and Practice*, Vol. 22, 494–508.

Narasimhan, K., & White, S. 2001. *Six Sigma: SPC and TQM in Manufacturing and Services20011Geoff Tennant.* Six Sigma: SPC and TQM in Manufacturing and Services . *Gower Publishing, 2000. 140 pp. , ISBN: ISBN 0-566-08374-4*. *The TQM Magazine*, Vol. 13.

Oestreich, T., & Chandler, N. 2015. Use the Gartner Business Analytics Compass to Drive Strategy. Retrieved from https://www.gartner.com/doc/3170223 [Accessed: June 11, 2016]

Ould, M. A. 2005. *Business Process Management: A Rigorous Approach*. BCS, The Chartered Institute.

Pérez-Castillo, R., De Guzmán, I. G. R., & Piattini, M. 2011. Business process archeology using MARBLE. *Information and Software Technology*, *53*(10), 1023–1044.

Pesic, M., Schonenberg, M. H., & Van Der Aalst, W. M. P. 2007. DECLARE: Full support for loosely-structured processes. *Proceedings – IEEE International Enterprise Distributed Object Computing Workshop, EDOC*, 287–298.

Ping, J., Chen, Y. S., Chen, B., & Howboldt, K. 2010. A robust statistical analysis approach for pollutant loadings in urban rivers. In *Journal of Environmental Informatics*, Vol. 16, 35–42. Springer.

Polato, M., Sperduti, A., Burattin, A., & De Leoni, M. 2014. Data-aware remaining time prediction of business process instances. *Proceedings of the International Joint Conference on Neural Networks*, 816–823.

Rebuge, Á., & Ferreira, D. R. 2012. Business process analysis in healthcare environments: A methodology based on process mining. *Information Systems*, *37*(2), 99–116.

Reddy, S. K., Barbas, A. S., Turley, R. S., Steel, J. L., Tsung, A., Marsh, J. W., Geller, D. A., & Clary, B. M. 2011. A standard definition of major hepatectomy: Resection of four or more liver segments. In *Hpb*, Vol. 13, 494–502. Eindhoven.

Redlich, D., Molka, T., Gilani, W., Blair, G., & Rashid, A. 2014. Constructs competition miner: Process control-flow discovery of BP-domain constructs. *Lecture Notes in*

Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), *8659 LNCS*, 134–150.

Rogge-Solti, A., & Kasneci, G. 2014. Temporal anomaly detection in business processes. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 8659 LNCS, 234–249. Springer.

Rogge-Solti, A., & Weske, M. 2013. Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 8274 LNCS, 389–403.

Rozinat, A., & Van Der Aalst, W. M. P. 2006. Decision mining in business processes. *BPM Center Report BPM-06-10*, *6*(10).

Rozinat, A., Van Der Aalst, W. M. P., & Weijters, A. J. M. M. 2010. *Process mining : conformance and extension*. *TU Eindhoven*.

Russell, N., Ter Hofstede, A. H. M., Van Der Aalst, W. M. P., & Mulyar, N. 2006. WORKFLOW CONTROL-FLOW PATTERNS A Revised View. *BPM Center Report*, *2*, 6–22.

Sadiq, S., & Governatori, G. 2015. Managing regulatory compliance in business processes. *Handbook on Business Process Management 2: Strategic Alignment, Governance, People and Culture, Second Edition*, 265–288.

Samo, R., Dewandono, R. D., Ahmad, T., Naufal, M. F., & Sinaga, F. 2015. Hybrid association rule learning and process mining for fraud detection. *IAENG International Journal of Computer Science*, *42*(2), 59–72.

Senderovich, A., Weidlich, M., Gal, A., & Mandelbaum, A. 2014a. Mining resource scheduling protocols. In S. Sadiq, P. Soffer, & H. Völzer (Eds.), *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 8659 LNCS, 200–216. Cham: Springer International Publishing.

Senderovich, A., Weidlich, M., Gal, A., & Mandelbaum, A. 2014b. Mining resource scheduling protocols. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8659 LNCS*, 200–216.

Senderovich, A., Weidlich, M., Gal, A., & Mandelbaum, A. 2014c. Queue mining - Predicting delays in service processes. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8484 LNCS*, 42–57.

Statista. 2017. Global business process management market forecast 2016-2021 | Statistic. Retrieved from https://www.statista.com/statistics/664859/worldwide-business-process-management-market-size/ [Accessed: July 11, 2017]

Steeman, W. 2013. BPI Challenge 2013, incidents. Ghent University. Dataset. Ghent University.

Sutrisnowati, R. A., Bae, H., Park, J., & Ha, B. H. 2013. Learning Bayesian network from event logs using mutual information test. *Proceedings – IEEE 6th International Conference on Service-Oriented Computing and Applications, SOCA 2013*, 356–360.

Szimanski, F., Ralha, G., Wagner, G., & Ferreira, D. R. 2013. Improving Business Process Models with Agent-Based Simulation and Process Mining. *Lecture Notes in Business Information Processing*, *147 LNBIP*, 124–138.

Tax, N., Verenich, I., La Rosa, M., & Dumas, M. 2017. Predictive business process monitoring with LSTM neural networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10253 LNCS*, 477–492.

Thomas, H., & James, E. 1990. *The New Industrial Engineering: Information Technology And Business Process Redesign*. *Management*.

Trkman, P. 2010. The critical success factors of business process management. *International Journal of Information Management*, *30*, 125–134.

Van Der Aalst, W. M. P. 2015. Business process simulation survival guide. *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*, 337–370.

Van Der Aalst, W. M. P., Adriansyah, A., & Van Dongen, B. F. 2012. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(2), 182–192.

Van Der Aalst, W. M. P., De Medeiros, A. K. A., & Weijters, A. J. M. M. 2005a. *Applications and Theory of Petri Nets 2005*. *Genetic Process Mining*, Vol. 3536.

Van Der Aalst, W. M. P., De Medeiros, A. K. A., & Weijters, A. J. M. M. 2005b. Genetic Process Mining. In *Applications and Theory of Petri Nets*, 48–69. Springer.

Van Der Aalst, W. M. P., Nakatumba, J., Rozinat, A., & Russell, N. 2010. Business Process Simulation : How to get it right ? *Handbook on Business Process Management*, 313–338.

Van Der Aalst, W. M. P., Pesic, M., & Schonenberg, M. H. 2009. Declarative workflows: Balancing between flexibility and support. *Computer Science – Research and Development*, *23*(2), 99–113.

Van Der Aalst, W. M. P., Rosemann, M., & Dumas, M. 2007. Deadline-based escalation in process-aware information systems. *Decision Support Systems*, *43*(2), 492–511.

Van Der Aalst, W. M. P., Schonenberg, M. H., & Song, M. 2011. Time prediction based on process mining. *Information Systems*, *36*(2), 450–475.

Van Der Aalst, W. M. P., & Weijters, A. J. M. M. 2004. Process mining: a research agenda. *Computers in Industry*, *53*(3), 231–244.

Van Der Aalst, W. M. P., Weijters, T., & Maruster, L. 2004. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, *16*(9), 1128–1142.

Van Der Werf, J. M. E. M., Van Dongen, B. F., Hurkens, C. A. J., & Serebrenik, A. 2009. Process discovery using integer linear programming. *Fundamenta Informaticae*, *94*(3–4), 387–412.

Van Dongen, B. F. 2012. BPI Challenge 2012. Eindhoven University of Technology.

Van Dongen, B. F. 2015. BPI Challenge 2015 Municipality 5. Eindhoven University of Technology.

Van Dongen, B. F., & Adriansyah, A. 2010. Process mining: Fuzzy clustering and performance visualization. *Lecture Notes in Business Information Processing*, *43 LNBIP*, 158–169.

Van Dongen, B. F., Crooy, R. A., & Van Der Aalst, W. M. P. 2008. Cycle time prediction: When will this case finally be finished? *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5331 LNCS*(PART 1), 319–336.

Van Dongen, B. F., De Medeiros, A. K. A., Verbeek, H. M. W., Weijters, A. J. M. M., & Van Der Aalst, W. M. P. 2005. The ProM Framework: A New Era in Process Mining Tool Support. In *Applications and Theory of Petri Nets 2005 SE - 25*, Vol. 3536, 444–454. Springer.

Van Dongen, B. F., De Medeiros, A. K. A., & Wen, L. 2009. Process mining: Overview and outlook of Petri net discovery algorithms. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 5460 LNCS, 225–242. Springer.

Van Dongen, B. F., & Van Der Aalst, W. M. P. 2005. A Meta Model for Process Mining Data. *{EMOI} - INTEROP'05, Enterprise Modelling and Ontologies for Interoperability, Proceedings of the Open Interop Workshop on Enterprise Modelling and Ontologies for Interoperability, Co-Located with CAiSE'05 Conference, Porto (Portugal), 13th-14th June 2005*, *160*, 309–320.

Verenich, I., Dumas, M., La Rosa, M., Maggi, F. M., & Di Francescomarino, C. 2016. Minimizing overprocessing waste in business processes via predictive activity ordering. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9694*, 186–202.

Vergidis, K., Tiwari, A., & Majeed, B. 2008. Business process analysis and optimization: beyond reengineering. *IEEE Transactions on Systems, Man, and Cybernetics*, *38*(1), 69–82.

Vossen, G. 2012. The process mining manifesto - An interview with Wil van der Aalst. In *Information Systems*, Vol. 37, 288–290.

Waddington, C. H. 1974. *Operations Research*. *Science*, Vol. 183.

Weber, I., Hoffmann, J., & Mendling, J. 2008. Semantic business process validation. *Proc of International Workshop on Semantic Business Process Management*, 22–36.

Weijters, A. J. M. M., & Ribeiro, J. T. S. 2011. Flexible heuristics miner (FHM). *IEEE SSCI 2011: Symposium Series on Computational Intelligence – CIDM 2011: 2011 IEEE Symposium on Computational Intelligence and Data Mining*, *334*(December), 310–317.

Weijters, A. J. M. M., Van Der Aalst, W. M. P., & De Medeiros, A. K. A. 2006. Process Mining with the Heuristics Miner Algorithm. *Technische Universiteit Eindhoven, Technical Report, WP*, *166*, 1–34.

Workflow Management Coalition. 1999. The Workflow Management Coalition Specification. *Workflow Management Coalition Terminology & Glossary*, *3.0*.

Wynn, M. T., Verbeek, H. M. W., Van Der Aalst, W. M. P., Ter Hofstede, A. H., & Edmond, D. 2009. Business process verification – finally a reality! *Business Process Management Journal*, *15*(1), 74–92.

Yi, M., & Filippidis, K. 2013. *BPSIM Standard*. *Workflow Management Coalition*.

Zhang, L. J. 2006. *SOA and Web services*. *Proceedings – ICWS 2006: 2006 IEEE International Conference on Web Services*.

Zhengxing Huang, X. L., & Huilong, D. 2011. Mining association rules to support resource allocation in business process management. *Expert Systems With Applications*, *38*(8), 9483–9490.

List of Scientific Publications by the Author on the Topic of the Dissertation

Papers in Reviewed Scientific Journals Indexed in Clarivate Analytics Web of Science

Savickas, T., Vasilecas, O. 2017. Decision support using belief network constructed from business process event log. *Informatica*. Vilnius: Vilnius University. 28(4), 583–598. http://dx.doi.org/10.15388/Informatica.2017.146.

Kalibatienė, D., Vasilecas, O.; Savickas, T., Vysockis, T., Bobrovs, V. 2016. A new approach on rule and context based dynamic business process simulation. (2016). *Baltic journal of modern computing (BJMC)*. Ryga: University of Latvia, 4(3), 408–419.

Papers in other editions

Vasilecas, O., Savickas, T., Lebedys, E. 2014. Directed acyclic graph extraction from event logs. *Information and Software Technologies: 20th International Conference, ICIST 2014, Druskininkai, Lithuania, October 9–10, 2014*. Berlin: Springer–Verlag, 172–181.

Vasilecas, O., Kalibatienė, D., Savickas, T., Šmaižys, A., Trinkūnas, J., Lebedys, E. 2014. Decision-making in information systems based on new development framework and business process mining. *Databases and information systems VIII: selected papers from the eleventh international Baltic conference on Databases and Information Systems (DB&IS)*. Amsterdam: IOS Press, 129–142.

Savickas, T., Vasilecas, O. 2015. Business process event log use for activity sequence analysis. *Electrical, Electronic and Information Sciences (eStream): proceedings of the 2015 Open conference, 21 April, 2015, Vilnius, Lithuania*. New York: IEEE, 1–4.

Vasilecas, O., Savickas, T., Normantas, K., Vysockis, T., Kalibatienė, Diana. 2016. A goal-oriented approach to dynamic business process simulation. *Frontiers in artificial intelligence and applications. Databases and information systems IX: conference proceedings of 12th International Baltic Conference on Databases and Information Systems 2016 (DB&IS 2016)*. Amsterdam: IOS Press, 143–154.

Savickas, T., Vasilecas, O. 2014. Business process event log transformation into Bayesian belief network. *Information systems development: Transforming organisations and society through information systems: proceedings of the 23rd international conference on information systems development (ISD2014 Croatia), September 2–4, 2014, Varaždin, Croatia*. Varaždin: University of Zagreb, Faculty of organization and informatics, 383–390.

Savickas, T., Vasilecas, O. 2014. Bayesian belief network application in process mining. *CompSysTech'14: proceedings of the 15th international conference on computer systems and technologies, Ruse, Bulgaria, June 27, 2014*. New York: ACM, 226–233.

Rusinaitė, T., Savickas, T., Vysockis, T., Vasilecas, O. 2016. Selection of activities in dynamic business process simulation. *Mokslas – Lietuvos ateitis: Elektronika ir elektrotechnika = Science – future of Lithuania: Electronics and electrical engineering*. Vilnius: Technika, 278–281.

Summary in Lithuanian

Įvadas

Problemos formulavimas

Verslo procesai (VP) yra vieni pagrindinių bet kurios organizacijos komponentų. Globali konkurencija skatina organizacijas nuolat tobulinti jose vykdomus VP. VP našumo didinimas bei egzistuojančių duomenų panaudojimas yra pagrindiniai veiksniai, lemiantys organizacijų sėkmę (Thomas & James, 1990; Trkman, 2010). Poreikis tobulinti VP lėmė operacijų tyrimais grindžiami metodų,VP modeliavimo bei imitacijos, procesų gavybos ir didžiųjų duomenų (angl. Big Data) analitikos atsiradimą. Visgi, VP analizei dažniausiai taikomi VP modeliavimo ir imitacijos metodai.

Įprastinių VP analizės metodų rezultatų kokybė dažnai netenkina, nes analizė remiasi nepatikima įvestimi. Taip yra todėl, kad metodų įvestis (pavyzdžiui, interviu su darbuotojais ar VP aprašymo dokumentai) gali neatspindėti tikrojo VP vykdymo. Taip pat realybėje vykstantys VP yra dinamiški ir stochastiški (Kellner *et al*., 1999; Van Der Aalst *et al*., 2010), nes jie nuolat kinta bei sprendimų priėmimo vykdyme dalyvauja žmonės. Tai lemia, kad ta pati VP įvestis negarantuoja tos pačios išvesties. Statiniai analizės metodai yra netinkami neraiškių, taikomų procesų vykdymo metu, žinių (angl. *inexplicit knowledge*) išgavimui. Tokiems atvejams viena iš tinkamų alternatyvų yra tikimybinių modelių naudojimas.

Pastaraisiais metais išsivystė procesų gavybos (angl. *Process Mining*) tyrimų kryptis, kurios taikymo sritis – įvykių žurnalų (angl. *Event Log*) panaudojimas verslo procesų analizei. Įvykių žurnaluose saugomi duomenys apie procesų vykdymą informacinėse sistemose (Griffeth *et al*., 2000). Procesų gavybos metodai sprendžia uždavinius, susijusius su: modelių išgavimu bei tobulinimu; modelių atitikties vertinimu lyginant žurnalą su realiai vykstančiais procesais; VP elgsenos prognozavimu ir kt. Šie metodai leidžia organizacijoms automatizuoti VP analizės užduotis. Kita vertus, šie metodai sprendžia tik mažas, specializuotas užduotis (Augusto *et al.*, 2017; Martin *et al*., 2015), o tai riboja platesnį jų taikymą.

Šioje disertacijoje tiriama įvykių žurnalų, aprašančių VP vykdymo istorinius duomenis, transformacija į tikimybinius modelius, siekiant įgalinti sprendimų paramą (angl. *decision support*) bei automatizuoti VP imitacinių modelių kūrimo procesą.

Darbo aktualumas

Visos organizacijos, norėdamos užtikrinti savo sėkmę, privalo taikyti VP valdymo metodus ir įrankius. Todėl, šių metodų ir įrankių vystymas įgauna vis didesnę svarbą (Statista, 2017).

Organizacijose sukuriamų duomenų panaudojimas *de facto* yra ta sritis, kuri sukuria didžiausią pridėtinę vertę. Duomenys gali būt panaudoti įvairių užduočių sprendimui, pavyzdžiui, rinkos analizei (Erevelles *et al*., 2016), sprendimų paramai (Kim *et al.*, 2014; Liu *et al*., 2012; Vasilecas, Kalibatiene, *et al*., 2014), sveikatos apsaugai (Groves *et al*., 2013).

Procesų gavyba leidžia naudoti istorinius duomenis sprendimų, vykdomų verslo procesuose, paramai. VP vykdymo duomenys gali suteikti didelę pridėtinę vertę, nes leidžia įvertinti egzistuojančių VP modelių bei jų vykdymo istorijos atitiktį, VP modelių išgavimą ar įgalinti sprendimų paramą, susijusią su vykdomais VP egzemplioriais (Griffeth *et al.*, 2000). Automatinis VP modelių išgavimas paspartina VP analizę, nes leidžia automatizuotai išgauti VP modelius iš istorinių duomenų ir sumažina žmogiškųjų išteklių poreikį (Augusto *et al*., 2017). Galiausiai, procesų gavyba taip pat leidžia automatizuoti sprendimų priėmimą, anomalijų VP vykdyme aptikimą ar VP vykdymo prognozavimą (Ceci *et al*., 2014; Rogge-Solti & Kasneci, 2014; Tax *et al.*, 2017). Visgi, nors ir egzistuoja daug metodų skirtų spręsti atskiras VP analizės ar imitacijos užduotis (Augusto *et al*., 2017; De Weerdt *et al*., 2012; Martin *et al*., 2015), vis dar nepilnai ištirtas kelias nuo duomenų apie VP vykdymą atsiradimo informacinėse sistemose iki VP analizės ir imitacijos.

Išvardintos problemos lėmė, kad šioje disertacijoje siekiama sukurti metodą, kuris galėtų iš įvykių žurnalo, aprašančio procesų elgseną, išgauti modelį, tinkamą sprendimų paramai ir VP imitacijai.

Tyrimo objektas

Disertacijos tyrimo objektas – VP prognozavimo ir elgsenos analizės, panaudojant išgautus VP (imitacinius) modelius iš įvykių žurnalo, procesas.

Darbo tikslas

Mokslinio darbo tikslas – patobulinti VP analizę bei imitaciją pasiūlant metodą, skirtą automatiškai išgauti tikimybinį verslo procesų modelį, ir sukurti imitacinį modelį iš įvykių žurnalo.

Darbo uždaviniai

Darbo tikslui pasiekti ir mokslinei problemai spręsti, darbe buvo iškelti šie uždaviniai:

1. Atlikti verslo procesų analizei skirtų procesų gavybos metodų analizę ir nustatyti egzistuojančių metodų trūkumus.
2. Išanalizuoti verslo procesų imitacijos metodus bei problemas, ribojančias jų taikymą.
3. Pasiūlyti metodą, skirtą iš žurnalo išgauti tikimybinį verslo proceso modelį, ir sukurti imitacinį modelį naudojant išgautą tikimybinį modelį.
4. Eksperimentiškai įvertinti pasiūlytą metodą, taikant dirbtinius ir realius duomenis.

Tyrimų metodika

Darbe taikyti šie tyrimų metodai:

1. Pažintinio tyrimo metodas taikytas mokslinio tyrimo objekto analizei, įsigilinant į problemas bei siekiant atlikti su tyrimų objektu susijusios literatūros analizę.
2. Konstruktyvinio tyrimo metodas taikytas konstruojant ir eksperimentiškai tiriant šioje disertacijoje siūlomus naujus metodus, skirtus išgauti VP tikimybinį modelį iš įvykių žurnalo ir transformuoti į imitacinius modelius, bei jų praktinį taikymą VP analizei. Pasiūlytas metodas buvo realizuotas, taikant C# programavimo kalbą, kaip savarankiškas prototipas tikimybinei analizei bei įskiepis egzistuojančioje VP imitacijos priemonėje DBPSim.

Darbo mokslinis naujumas

Darbo mokslinis naujumas pagrįstas šiais rezultatais:

1. Pasiūlytas metodas leidžia iš įvykių žurnalo išgauti Bajeso tikimybinį modelį (angl. *Bayes Belief Network*). Metodas remiasi nauju algoritmu, kuris leidžia išgauti kryptinį beciklį grafą (angl. *Directed Acyclic Graph*) iš įvykių žurnalo. Pasiūlytas metodas įgalina sprendimų paramą ir leidžia prognozuoti vykstančio proceso egzemplioriaus tolimesnius įvykius.
2. Pasiūlytas naujas metodas leidžia automatizuotai sukurti pradinius VP imitacinius modelius. Metodas transformuoja Bajeso tikimybinį modelį į dinaminių verslo procesų imitacijos (angl. *Dynamic Business Process Simuliation*) modelį.

Darbo rezultatų praktinė reikšmė

Šiame darbe pasiūlytas naujas metodas Bajeso tikimybinio modelio išgavimui gali būti taikomas sprendimų paramai, t. y. aptikti vykdomo VP egzemplioriaus anomalijas bei prognozuoti VP vykdymą (įskaitant būsimus duomenis). Pasiūlytas metodas leidžia analitikams sukurti pradinius verslo procesų modelius, kurie gali būti taikomi VP variantų (angl. *what-if*) analizei. Metodas leidžia sumažinti VP imitacinių modelių kūrimo laiko sąnaudas.

Pasiūlytas metodas buvo įgyvendintas prototipuose – tikimybinio modelio išgavimas įgyvendintas naujame prototipiniame įrankyje BBNGs, o šis įrankis buvo integruotas į dinaminių verslo procesų imitacijos priemonę DBPSim. Prototipo veikimui reikalingas tik korektiškas plačiai taikomo XES formato failas, todėl, su mažais patobulinimais, sukurtas metodas galėtų būti taikomas praktikoje.

Ginamieji teiginiai

Šios disertacijos ginamieji teiginiai yra:

1. Pasiūlytas Bajeso tikimybinio modelio išgavimo iš įvykių žurnalo metodas geba panaikinti ciklus kryptinio beciklio grafo išgavimo metu. Taip pat metodas geba įvertinti VP egzemplioriaus įvykių tikimybę 63–98 % tikslumu ir prognozuoti toliau įvyksiančius verslo proceso egzemplioriaus įvykius 72–87 % tikslumu naudotiems eksperimentiniams žurnalams.
2. Pasiūlytas metodas geba automatizuotai sukurti dinaminių VP imitacijos modelius, o jų vykdymas pasiekia 58–98 % kaina paremtą atitiktį (angl. *cost-based fitness*), kai įvykių žurnalas, gautas imitacijos metu, lyginamas su pradiniu įvykių žurnalu.

Darbo rezultatų aprobavimas

Disertacijos autorius paskelbė 9 mokslines publikacijas disertacijos tema, iš kurių: 2 publikuotos žurnaluose, įtrauktuose į Clarivate Analytics (buv. *Thomson Reuters*) *Web of Science* duomenų bazę, 7 – mokslinių konferencijų pranešimų rinkiniuose. Moksliniai rezultatai buvo pristatyti 4 mokslinėse konferencijose:

- *20th International Conference on Information and Software Technologies* (ICIST 2014). 2014 m. spalio 9–10 d., Druskininkai, Lietuva.
- *Electrical, Electronic and Information Sciences (eStream).* 2015 m. Balandžio 21 d., Vilnius, Lietuva.
- *23rd International Conference on Information Systems Development* (ISD2014 *Croatia*). 2014 m. rugsėjo 2–4 d., Varaždin, Kroatija.
- *Data Analysis Methods for Software Systems* (DAMSS). 2016 m. gruodžio 1–3 d, Druskininkai, Lietuva.

Disertacijos struktūra

Disertacija yra sudaryta iš įvado, trijų pagrindinių skyrių, bendrųjų išvadų, šaltinių sąrašo, disertacijos autoriaus publikacijų sąrašo bei lietuviškos santraukos. Disertacijos apimtis: 124 puslapiai, 20 formulių, 2 algoritmai, 23 paveikslai ir 12 lentelių.

1. Verslo procesų analizės metodų tyrimas

Verslo procesai yra vykdomi visose organizacijose, todėl konkurencija skatina nuolatinį VP tobulinimą. Nuolatinis procesų našumo didinimas bei žinių (per-)panaudojimas lemia organizacijų sėkmę (Thomas & James, 1990; Trkman, 2010). Poreikis tobulinti VP lėmė verslo procesų tobulinimo būdų atsiradimą.

VP gali būti labai paprasti ir juose gali vykti vos keletas veiklų, pvz., kasininkės darbas, arba VP gali būti labai sudėtingi ir trukti ilgai, o jų vykdyme gali dalyvauti daug skirtingų žmonių bei sistemų. VP modeliavimas ir valdymas yra sudėtingas, nes (Kellner *et al*., 1999):

- procesai yra nenuspėjami. Viena vertus, procesai gali būti sunkiai suvaldomi dėl žmogiškųjų išteklių elgsenos (Van Der Aalst *et al*., 2010). Kita vertus, patys procesai gali sąveikauti su kitais procesais, kurių neįmanoma kontroliuoti, nes jie yra vykdomi organizacijos išorėje (Zhang, 2006).
- procesai nuolat kinta (Bose *et al*., 2014) ir nauja elgsena gali atsirasti be atitinkamos dokumentacijos ar aprašymo.
- procesuose egzistuoja sudėtingi grįžtamojo ryšio mechanizmai. Tai lemia, kad VP pradžioje vykdomos veiklos nulemia toliau vykdomų veiklų seką ir rezultatus. Tokie grįžtamojo ryšio mechanizmai dažnai yra nežinomi ir neapibrėžti modeliuose. Taip pat siekiant supaprastinti proceso aprašymą, šie grįžtamojo ryšio mechanizmai gali būti iš viso nemodeliuojami.

Šiuolaikinės organizacijos didžiąją savo duomenų dalį saugo įvairiose skaitmeninėse talpyklose, pvz., duomenų bazių valdymo sistemose, skaitmeniniuose dokumentuose ir kt.

Organizacijose vykdomų VP analizė gali būti atliekama taikant procesų gavybos (PM) metodus (Vossen, 2012). PM metodai naudoja įvykių žurnalus, kuriuose esantys duomenys aprašo istorinį verslo procesų vykdymą (VP elgseną). Žurnaluose saugomas procesų egzempliorių sąrašas, įvykių sekų sąrašas bei įvykius aprašantys duomenys. VP modelių išgavimui egzistuoja daug įvairių metodų, tačiau šie metodai skirti specializuotų užduočių sprendimui. Pirmiausia, metodai gali išgauti VP modelius tam tikra modeliavimo kalba, pavyzdžiui, Petri tinklais (De Medeiros, 2006; Leemans *et al*., 2013, 2014; Rebuge & Ferreira, 2012; Van Der Aalst *et al*., 2004; Van Der Werf *et al*., 2009; Weijters & Ribeiro, 2011; Weijters *et al*., 2006) ar deklaratyvių procesų modeliavimo kalbomis (Di Ciccio *et al*., 2016; Maggi *et al*., 2012; Maggi, Dumas, García-Bañuelos, & Montali, 2013b; Pesic *et al*., 2007). Taip pat PM metodai gali būti skirti išgauti VP modelius, tenkinančius ribojimus, pavyzdžiui, atspindėti veiklų priežastinį ryšį taikant priklausomybės tinklus (angl. *causal network*) (Weijters *et al*., 2006), garantuoti tinkamus (angl. *fitting*) ar pagrįstus (angl. *sound*) modelius.

VP modelių teisingumą gali apskaičiuoti Vertinimo metodai. Jie įvertinta ar VP modelyje esančios veiklų sekos atitinka (angl. *fits*) įvykių žurnaluose esančias įvykių sekas. Atitikties (angl. *fitness*) metrikos kiekybiškai įvertina kiek elgsena, galima VP modelyje, atitinka elgseną, pastebėtą įvykių žurnale. Jei modelis leidžia platesnę elgseną nei pastebima įvykių žurnale, tai laikoma, kad modelis yra nepakankamai atitinkantis (angl. *underfitting*) ir, atvirkščiai, kai modelis leidžia ne visą elgseną, kuri matoma žurnale, modelis laikomas pernelyg atitinkančiu (angl. *overfitting*).

PM metodai tinka ne tik VP modelių išgavimui bei vertinimui, bet ir sprendimų paramai ar VP elgsenos parametrų nustatymui. Šie metodai geba nustatyti tuos VP parametrus, kurie leistų padidinti VP našumą ar patobulinti VP modelius. Pavyzdžiui, VP elgsenos parametrų išgavimo metodai gali nustatyti išteklių priskyrimo taisykles (Ferreira & Alves, 2012; Ly *et al*., 2006; Senderovich, Weidlich, Gal, & Mandelbaum, 2014b; Zhengxing Huang & Huilong, 2011). Viena iš pagrindinių sprendimų paramos sričių, kurioms tinkami PM metodai, yra VP laiko parametrų valdymas, pvz., VP egzemplioriaus trukmės nustatymas (Polato *et al*., 2014; Rogge-Solti & Weske, 2013; Van Dongen *et al*., 2008). Siekiant iš anksto numatyti galimas VP egzempliorių vykdymo problemas, būtina iš anksto žinoti apie jo tikėtiną baigtį. Tai galima atlikti panaudojant įvykių žurnalą VP prognozavimui (Ceci *et al*., 2014; Cook & Wolf, 1998; Ferreira *et al*., 2007; Folino *et al*., 2014; Polato *et al*., 2014; Rogge-Solti & Weske, 2013; Tax *et al*., 2017). Visgi, visi rasti metodai akcentuoja vieno parametro, pavyzdžiui, laiko ar trukmės, vertinimą. Leka neaišku, kiek šie metodai yra tinkami prognozuoti kitus dalykinės srities duomenis.

Imitacija yra procesas, skirtas sukurti tam tikros probleminės srities modelį ir imituoti jos elgseną neturint įtakos pačiai probleminei sričiai. Taip pat imitacija dažniausiai taikoma siekiant patobulinti sistemos veikimą ar numatyti, kaip sistema veiks, kai bus įgyvendinta. Galiausiai, imitacija leidžia aptikti galimas modelių klaidas, nustatyti „butelio kaklelius“ ar išteklių perkrovą. Imitacija gali būti deterministinė (ta pati įvestis garantuoja tą pačią išvestį) arba stochastinė (ta pati įvestis negarantuoja tos pačios išvesties). Stochastinės imitacijos pavyzdys yra *Monte-Carlo* imitacija.

Pagrindinės VP imitacijos problemos:

- Įprastos VP modeliavimo kalbos yra nukreiptos į kontrolės srauto modeliavimą, bet ne į veiklų elgseną, todėl įprasti VP modeliai netinka detalios VP elgsenos analizei, o norint atlikti imitaciją reikia kurti specializuotus VP imitacinius modelius.
- Nėra standartizuotų VP imitacijos sprendimų, o įrankiuose naudojamos nuosavos VP imitacinio modeliavimo kalbos. Tik 2013 m. atsirado pirmasis VP imitacijos modeliavimo standartas DBPSim (Yi & Filippidis, 2013), leidžiantis papildyti BPMN modelius imitacijos parametrais. Šis standartas nėra plačiai taikomas.
- Įprasti VP modeliai neturi pakankamai informacijos apie VP elgseną (pvz., kaip žmonės sąveikauja (Yi & Filippidis, 2013)) arba juose modeliuojama supaprastinta elgsena (Fadel *et al*., 1994; Vasilecas, Normantas, *et al*., 2016). Detaliai analizei reikia naudoti dalykinės srities duomenis.

Dinaminių verslo procesų imitacija (Vasilecas *et al*., 2015) leidžia atlikti procesų imitaciją, kurioje veiklų vykdymas ir jose esantys duomenys yra detaliai imituojami (Vasilecas, Normantas, *et al*., 2016).

PM metodų taikymas leidžia paspartinti modelių kūrimą bei padidinti modelių tikslumą, nes naudojami istoriniai VP vykdymo duomenys. Kaip rodo Martino atlikta VP imitacijai tinkamų sprendimų analizė (Martin *et al*., 2016), egzistuoja daug metodų, tinkamų VP imitacijos uždavinių sprendimui, tačiau nėra tyrimų, įrodančių šių metodų efektyvumą VP imitacijoje.

2. Bajeso tikimybinio modelio ir imitacinio modelio išgavimo iš įvykių žurnalo metodas

VP tobulinimui būtina jų analizė. Ši analizė skirta įvertinti VP pakeitimų pasekmes ar nustatyti tobulintinas vietas. Priklausomai nuo siekiamų rezultatų, analizuoti galima istorinę VP elgseną, dabartinę VP būseną arba tolimesnį VP vykdymą.

Šiame skyriuje pristatomas siūlomas metodas, skirtas išgauti Bajeso tikimybinį modelį (BBN) iš įvykių žurnalo ir jį panaudoti imitacinio modelio kūrimui (S2.1 pav.). Skyriaus turinys remiasi autoriaus anksčiau publikuotais tyrimais (Kalibatiene *et al*., 2016; Savickas & Vasilecas, 2014, 2015, 2017; Vasilecas, Kalibatiene, *et al*., 2014; Vasilecas *et al*., 2015).

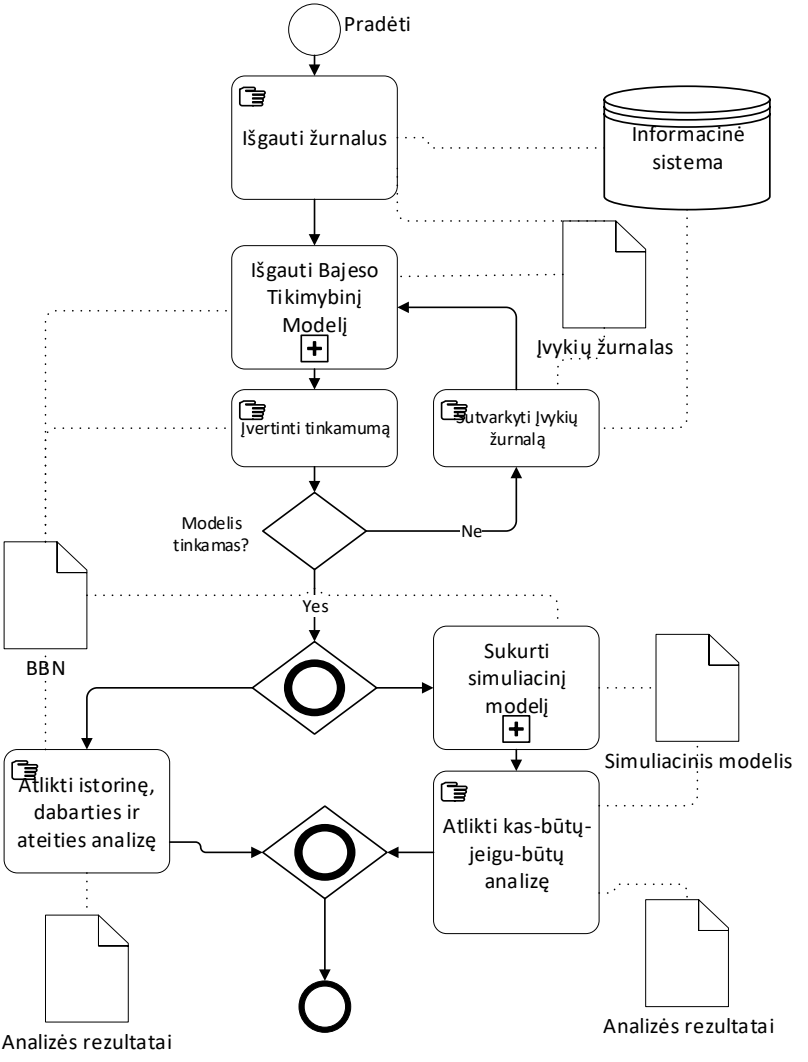
Dvi pagrindinės problemos, sunkinančios VP analizės taikymą, yra tai, kad nenaudojamos IS egzistuojančios žinios, bei tai, kad dalykinės srities modeliavimas yra daug laiko reikalaujantis uždavinys. Siekiant išspręsti minėtas problemas, būtina pakartotinai panaudoti IS egzistuojančias žinias apie tai, kaip istoriškai vyko VP. Taip pat būtina automatizuoti dalykinės srities modeliavimą ir imitaciją. Atlikus šias užduotis, būtų pagreitinta VP analizė bei pagerinti analizės rezultatai.

Siūlomas sprendimas prasideda nuo egzistuojančių žinių pakartotinio panaudojimo. Visi VP, kurie yra automatizuoti IS, palieka savo vykdymo pėdsaką duomenų bazių valdymo sistemose ar kituose skaitmeniniuose šaltiniuose. Šių VP vykdymo istorinių duomenų (įvykių žurnalų) panaudojimas VP analizei nėra nauja sritis – tai yra Procesų gavybos (PM) pagrindas. Siūlomas sprendimas taip pat naudoja įvykių žurnalus kaip įvestį. Šiame darbe įvykio žurnalo šaltinio formatas ignoruojamas, nes duomenų išgavimas kiekvienu atveju yra unikalus ir priklauso nuo IS įgyvendinimo. Šis uždavinys paliekamas analitikui.

Antra siūlomo sprendimo dalis padeda spręsti dalykinės srities modeliavimo problemą. Dalykinės srities modelis turi gebėti: atvaizduoti VP kontrolės srautą; atsižvelgti į tai, kad VP modelis naudoja ir generuoja įvairius duomenis; atsižvelgti į tai, kad VP iš prigimties yra dinamiški; atsižvelgti į grįžtamojo ryšio mechanizmus.

Statiniai analizės metodai gali atvaizduoti kontrolės srautą ir duomenis, tačiau jie neatsižvelgia į grįžtamąjį ryšį bei VP dinamiškumą. Dirbtinio intelekto sprendimai, pvz., neuroniniai tinklai ar paramos vektorių mašinos, geba atsižvelgti į stochastiškumą ir grįžtamojo ryšio mechanizmus, tačiau negali paaiškinti savo sprendimų ar atvaizduoti kontrolės srauto. Geriausiai keliamus reikalavimus tenkina tikimybiniai modeliai, kurie remiasi grafų teorija. Bajeso tikimybiniai modeliai (BBN) yra tinkamiausi, nes jų kryptinis beciklis grafas (DAG) leidžia atvaizduoti kontrolės srautą, tikimybinių skirstinių lentelės (CPT) atvaizduoja generuojamus ir naudojamus duomenis, o išvedimo mechanizmai leidžia atsižvelgti į grįžtamąjį ryšį ir VP stochastiškumą. BBN turi dvi pagrindines dalis – DAG ir

CPT: DAG yra skirti atvaizduoti įvykių sąlyginę priklausomybę, o CPT naudojamos aprašyti tam tikro mazgo duomenų tikimybes.



S2.1 pav. Siūlomo metodo struktūrinė schema

DAG kūrimo sub-proceso uždavinys yra iš įvykių žurnalo sukurti grafą, kuriame kiekvienas įvykis atspindimas mazgu, o kontrolės srautas tarp veiklų – kraštinėmis. Pagrindinis DAG ribojimas yra tai, kad, priešingai nei VP, DAG negali turėti ciklų. Metodas, grafo kūrimo metu, iteruoja pro įvykius žurnale ir, pastebėjęs ciklą, jį sudarančius mazgus padaro nepriklausomus arba transformuoja į ciklą apimantį mazgą (Vasilecas, Savickas,

et al., 2014). DAG kūrimo metu papildomai panaudojama dažnių matrica, kurios taikymas leidžia nustatyti perėjimus tarp įvykių ar jų tarpusavio priklausomybę. Dažnių matrica remiasi Euristinio išgavėjo naudojamu priežastiniu tinklu (angl. *Causal Network*) (Weijters *et al*., 2006), kuris buvo išplėstas siekiant įgalinti įvykių nepriklausomybės nustatymą.

CPT išgavimas yra antras BBN kūrimo sub-procesas. CPT kuriamos pakartotinai iteruojant pro įvykių žurnalą ir kiekvienam mazgo duomenų rinkiniui, priklausomai nuo ankstesnių įvykių duomenų, konstruojant tikimybinių skirstinių lenteles. Baigus iteruoti pro įvykius žurnale, kiekvienam mazgui priskiriama tikimybinių skirstinių lentelė.

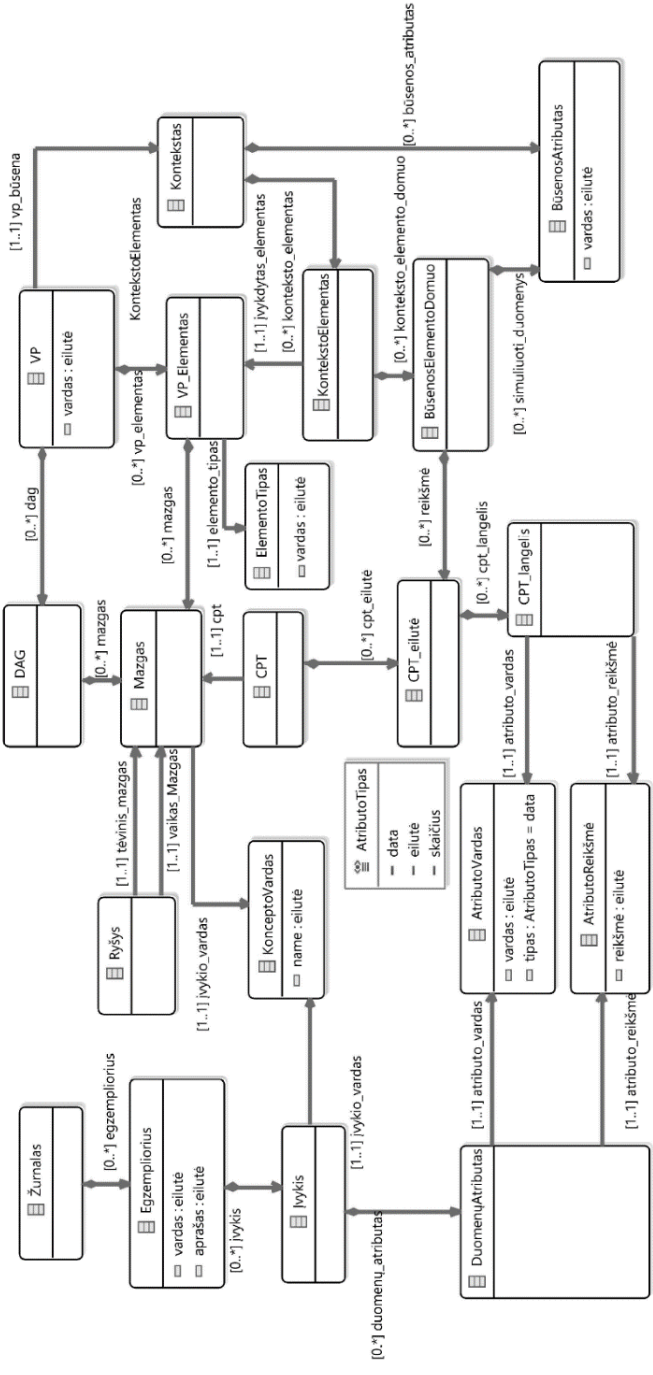
Norint vykdyti VP imitaciją, pirmiausia reikia turėti VP imitacinį modelį. Standartiniai VP imitaciniai modeliai yra statiniai (t. y. visi elementai ir kontrolės srautai yra iš anksto numatyti), todėl ir imituojami VP būna statiniai, o imitacija teikia ribotus rezultatus. Taip pat, įprasti imitaciniai modeliai nenaudoja detalių duomenų, kuriamų veiklų vykdymo metu ar naudojamų sprendimų priėmimui. Dinaminių verslo procesų imitacija (DBPS) leidžia detaliai imituoti VP, nes VP kontrolės srautas nėra iš anksto numatytas bei daroma prielaida, kad veiklų iniciaciją nustato predikatų taisyklės. Be to, DBPS veiklų vykdymui ar sprendimams leidžia naudoti detalius kontekstinius duomenis (Kalibatiene *et al*., 2016).

Siūlomas sprendimas transformuoja BBN į DBPS imitacinius modelius. Transformacijos procesas yra sudarytas iš šių žingsnių:

1. Išgaunamas BBN iš įvykių žurnalo.
2. Sukuriamas DBPS modelis panaudojant BBN.
3. Sukombinuojamas BBN su perėjimų sistema (Van Der Aalst *et al*., 2011), kad įgalintų perėjimų tikimybių išvedimą.
4. Imitacinis modelis tobulinamas rankiniu būdu pagal analitiko poreikius.

DBPS modelio kūrimo metu kiekvienas įvykis transformuojamas į veiklą, kurios aktyvacijos taisyklė yra tokia – „*jei pagal esamą proceso būseną įvykio tikimybė yra didžiausia, veikla turi būti atliekama“*. Vykdoma veikla keičia VP būseną bei sugeneruoja atsitiktinius duomenis panaudojant CPT rinkinius. Sugeneruoto duomenų atributo *duration* reikšmė nustato veiklos vykdymo trukmę. Sukurtas DBPS imitacinis modelis yra tik pradinis ir imituoja žurnalo elgseną. Todėl, siekiant spręsti specializuotą analizės uždavinį, analitikai gali papildyti ar pakeisti imitacinį modelį, papildyti ar pakeisti modeliuojamų veiklų turinį, pridėti papildomas aktyvacijos taisykles.

Ryšys tarp žurnalo, BBN ir imitacinio modelio elementų yra pateiktas S2.2 paveiksle. Šiame paveiksle pateikta klasių diagrama, kurioje nurodyti įvykių žurnalų elementai (*Žurnalas, Egzempliorius, Įvykis, DuomenųAtributas, KonceptoVardas, AtributoVarads, AtributoReikšmė*), tikimybinio modelio elementai (*DAG, Mazgas, CPT, CPT_eilutė, CPT_langelis*) bei imitacinio modelio elementai (*VP, VP_elementas, KontekstoElementas ir kt.*). Ryšiai tarp klasių nurodo kokie žurnalo elementai naudojami tikimybinio ir imitacinių modelių kūrimui.



S2.2 pav. Ryšys tarp įvykių žurnalo, Bajeso tikimybinio modelio ir imitacinio modelio elementų

3. Pasiūlyto metodo eksperimentiniai tyrimai

2 skyriuje pristatytas siūlomas metodas susideda iš kelių žingsnių, kurie turi būti eksperimentiškai ištirti. Pagrindinė metodo įvestis yra įvykių žurnalas, todėl, siekiant tinkamai ištirti pasiūlytą metodą, būtina pasirinkti tokius įvykių žurnalus, kurie padengtų įvairią procesų elgseną ir leistų pakartoti eksperimentus. Dėl šių priežasčių buvo pasirinkti 5 skirtingi įvykių žurnalai. Pirmasis žurnalas yra dirbtinis viešai prieinamas žurnalas, aprašantis draudiminio įvykio procesą (Van Der Aalst *et al*., 2007), kuris buvo pasirinktas dėl savo paprastumo ir mažo triukšmo. Kiti trys pasirinkti žurnalai – viešai prieinami *Business Process Intelligence* konferencijos varžybose naudoti įvykių žurnalai: BPI'12 – žurnalas su Olandijos Finansinės institucijos paskolos prašymo proceso duomenimis (Van Dongen, 2012), BPI'13 – žurnalas su automobilių gamintojo incidentų valdymo proceso duomenimis (Steeman, 2013) bei BPI'15 – įvykių žurnalas su savivaldybės statybos leidimo išdavimo proceso duomenimis (Van Dongen, 2015). Paskutinis pasirinktas žurnalas – studentų įsakymų priėmimo proceso istoriniai duomenys (EIMSD). Studentų įsakymų priėmimo procesas yra dinamiškas, nes veiksmų įvykių seka nėra iš anksto aiškiai apibrėžta. Pasirinktų žurnalų savybės pateiktos S3.1 lentelėje.

S3.1 lentelė. Eksperimentams parinktų įvykių žurnalų savybės

Žurnalo pavadinimas	Egzemplioriai, vnt.	Unikalūs įvykiai, vnt.	Iš viso įvykių, vnt.	Atributų kiekis, vnt.
Synthetic log	3 512	9	20 339	2–6
BPI 12	13 087	36	262 200	3–4
BPI 13	7 554	13	65 535	9
BPI 15	1 156	289	59 083	12
EIMSD	2 774	63	21 392	6

Eksperimentų formulavimas vykdomas po eksperimentų duomenų parinkimo. Pirmoji metodo dalis sukuria BBN iš įvykių žurnalo. BBN kūrimas susideda iš dviejų žingsnių – DAG ir CPT kūrimo. DAG paskirtis yra atvaizduoti įvykių sekas (su panaikintais ciklais), todėl pirmiausia įvertinamas DAG kūrimas. Kadangi DAG yra grafas ir jis skirtas tik vizualizacijai, todėl jo vertinimas irgi yra vizualus – atliekamas grafinis palyginimas tarp grafo, išgauto taikant pasiūlytą metodą, bei grafų išgautų taikant alternatyvius metodus, pavyzdžiui, euristinio išgavėjo (angl. *Heuristic Miner*) (Weijters *et al*., 2006).

Nėra prasmės atskirai vertinti CPT, nes jos naudojamos duomenų tikimybių išvedimui. Tolesnis eksperimentas yra BBN veikimo eksperimentinis tyrimas, kuriame vertinamas tikimybių išvedimo teisingumas. Vertinimas atliekamas, panaudojant skirtingus žurnalus, tokiais žingsniais:

1. Žurnalas išskaidomas į dvi dalis – 90 % ir 10 %.
2. 90 % dalis naudojama išgauti ir paruošti BBN.
3. Likę 10 % naudojami eksperimentuose.
4. Eksperimentas kartojamas dar dešimt kartų, panaudojant skirtingas žurnalo dalis, tam, kad pasiekti *k* blokų kryžminį tikrinimą (k = 10).

Tikimybių vertinimo eksperimentai atliekami imituojant VP vykdymą. Eksperimento metu iteruojama pro žurnale esančius įvykius ir kuriama tarpinė proceso būsena $l^{state}(\sigma)$, kur σ yra šiuo metu pasiekta egzemplioriaus dalis žurnale. Žinant koks buvo paskutinis įvykis su būsena $l^{event}(\sigma(k))$, galima apskaičiuoti $P\left(l^{event}(\sigma(k))|l^{state}\left(hd^{k-1}(\sigma)\right)\right)$, t. y. sekančio egzemplioriaus įvykio tikimybę, žinant iki šiol įvykusius egzemplioriaus įvykius. Tikimybės skaičiavimas atliekamas tik tada, kai $|\sigma| > 0$, t. y. kai būsenoje yra bent vienas įvykis. Pirmasis įvykis ignoruojamas, nes prieš pirmą įvykį proceso būsena yra tuščia, todėl ir išvesta tikimybė nevertina metodo.

Tikimybių išvedimo tikslumo vertinimui atliekami šie eksperimentai:

1. Skaičiuojama vidutinė įvykių tikimybė bei standartinis nuokrypis.
2. Vertinama, kaip gerai metodas geba prognozuoti proceso egzemplioriaus ateitį:
   a) su skirtingais atributų rinkinių dydžiais – vertinama kaip duomenų kiekis daro įtaką prognozavimo efektyvumui;
   b) su skirtingu žurnalų sudėtingumu – vertinamas bendras metodo tinkamumas įvairaus sudėtingumo procesams.

Verslo proceso egzemplioriaus įvykio prognozavimas gali būti:

- teisingas (angl. Correct), kai kitas įvykis žurnale yra tas, kuris buvo prognozuotas.
- teisingas intervale (angl. correct interval prediction), kai kitas įvykis žurnale nėra tas, kuris buvo spėtas, tačiau įvykis pasirodė žurnale vėliau tame pačiame egzemplioriuje. Jei teisingos prognozės intervale kartojasi tam pačiam egzemplioriaus įvykiui, prognozės rezultatas praleidžiamas.
- neteisingas, kai prognozuotas įvykis proceso egzemplioriuje neįvyko.

Paskutinis žingsnis yra imitacijos eksperimentinis tyrimas. Pasiūlyto metodo imitacijos tikslumas vertinamas šiais žingsniais:

1. Įkeliamas įvykių žurnalas (SEL) į DBPSim.
2. Taikant metodą, sukuriamas imitacinis modelis (naudojant įvykių žurnalą).
3. Atliekama bent 250 VP egzempliorių imitacija ir sukuriamas rezultatų įvykių žurnalas (GEL).
4. Atliekamas atitikties vertinimas tarp SEL ir GEL:
   a) išgaunami procesų modeliai Petri tinklo pavidalu iš sugeneruoto įvykių žurnalo (GMM) ir pradinio įvykių žurnalo (SMM), taikant Indukcinį išgavėją (angl. Inductive Miner) (Leemans *et al*., 2013);
   b) atliekamas atitikties vertinimas tarp SEL ir GMM bei tarp GEL ir SMM taikant Pakartojimo metodą (Van Der Aalst *et al*., 2012).
5. Vertinami rezultatai.

Pakartojimo metodas ignoruoja įvykių duomenis atitikties vertinimo metu. Duomenų atitiktis ir neturi būti vertinama, nes duomenys generuojami pseudo-atsitiktinai, atsižvelgiant į istorinį VP vykdymą.

DAG išgavimo vertinimui, įvykių žurnalai buvo įkelti į BBNGs ir iš įvykių žurnalų buvo išgauti DAG, atvaizduojantys verslo procesus. Tie patys žurnalai buvo įkelti į plačiai taikomą procesų gavybos priemonę PRoM (Van Dongen et al., 2005) ir, pritaikius PRoM esančius procesų modelių išgavimo metodus (Van Der Aalst et al., 2005b, 2004; Van Dongen et al., 2009), buvo išgauti lyginamieji VP modeliai. Išgauti VP modeliai buvo

vizualiai panašūs į pasiūlyto metodo išgautus modelius, tačiau grafai, išgauti taikant siūlomą metodą, neturėjo ciklų.

Antrasis eksperimentas – BBN taikymo tikimybių išvedimui tyrimas. Eksperimento metu buvo panaudota 17 750 procesų egzempliorių ir iš viso atliktas 232 861 įvykių tikimybių išvedimas. Iš viso atmesti 74 168 išvedimai, kurie buvo: pirmo proceso egzemplioriaus įvykio; anomalijos ($P(D|H) = 0$); nežinomi. Nežinomi įvykiai yra tokie, kurie nepasitaikė apmokymo duomenyse. Gauti išvedimo rezultatai pateikti S3.2 lentelėje.

S3.2 lentelė. Tikimybių išvedimo rezultatai su *k* blokų kryžminiu vertinimu ir k = 10

Žurnalo pavadinimas	Panaudoti išvedimai, vnt.	Visi išvedimai/ Iš viso įvykių žurnale, vnt.	Pastebėti unikalūs įvykiai/ Unikalūs įvykiai žurnale, vnt.	Vidutinė tikimybė, %
Synthetic	10 679	16 782/20 339	8/9	78,71±22,73
BPI'12	122 720	147 255/262 200	34/36	63,1±14,26
BPI'15	11 236	53 324/59 083	35/289	98,16±16,36
EIDSM	14 058	15 500	58/63	62,66±10,25

Geriausi rezultatai buvo Synthetic žurnalo atveju – vidutinė išvesta tikimybė siekė 78,71 % ir net 4 įvykiai turėjo vidutinę išvestą tikimybę didesnę nei 99 % su <1 % standartiniu nuokrypiu. Kitų įvykių išvestos tikimybės svyravo nuo 30 % iki 85 % ir standartinis nuokrypis svyravo tarp ±36 % ir ±50 %.

Kiti procesai yra sudėtingesni ir tai atsispindi gautuose rezultatuose. BPI'12 žurnalo įvykiai turėjo tik 3–4 duomenų atributus, todėl nėra aišku, ar įvykių duomenys aprašo tarpusavio priklausomybę. Gauti rezultatai rodo, kad vidutinė prognozuota tikimybė buvo 51 %, bet 10 iš 36 įvykių vidutinė išvedimo tikimybė buvo didesnė nei 80 %.

Mažiausiai panaudotų įvykių buvo iš BPI'15 žurnalo (11 236 iš 53 324), tačiau pats procesas yra sudėtingiausias, nes jame iš viso yra 289 galimi įvykiai ir tik 1 156 egzemplioriai įvykių žurnale. Mažas duomenų kiekis nulėmė, kad BBN apmokymui neužteko duomenų ir dėl to nukentėjo tikimybių išvedimo rezultatai. Nepaisant to, vidutinė išvesta tikimybė siekė 99 %. Be to, 33 iš 35 įvykių, kurių tikimybių išvedimai panaudoti rezultatuose, vidutinė išvesta tikimybė viršijo 80 %.

Trečiasis eksperimentų etapas – BBN tinkamumo prognozuoti VP vykdymą vertinimas. Pirmiausia buvo tiriama, kaip duomenų atributų kiekis žurnale daro įtaką prognozavimo kokybei. Tam buvo panaudotas Synthetic žurnalas. Eksperimento metu, buvo atliktas tikimybių išvedimas su pilnu duomenų atributų rinkiniu, o po to su sumažintu duomenų atributų rinkiniu. Rezultatai palyginti su atsitiktiniu spėjimu, kuriame kiekvienas kontrolės srauto ėjimas pasirenkamas atsitiktinai, pagal tai kaip istoriškai dažnai buvo einama skirtingais keliais.

Eksperimentai parodė, kad BBN demonstruoja geresnius rezultatus nei atsitiktinis kontrolės srauto spėjimas, remiantis statistiniais svoriais atskiriems keliams. Be to, duomenų atributų pridėjimas pagerina gaunamus rezultatus. Visgi, reikalingi papildomi tyrimai, kurie nustatytų kokios duomenų savybės daro įtaką prognozės tikslumui.

Antras prognozavimo vertinimas atliktas atkartojant žurnalų vykdymą ir bandant prognozuoti kitus VP egzemplioriaus įvykius. Gauti rezultatai pateikti S3.3 lentelėje.

S3.3 lentelė. Prognozavimo rezultatai su *k* blokų kryžminiu vertinimu ir k = 10

Žurnalas	Teisingai, %	Teisingai intervale, %	Neteisingai, %	Praleista, %
Synthetic	84,49±18,66	2,52±7,29	5,74±8,06	7,25±19,67
BPI'12	73,49±31,36	3,61±11,31	13,7±18,86	9,2±25,04
BPI'15	53,01±31,54	21,36±23,27	18,24±22,55	7,17±25,01
EIMSD	67,6±32,54	4,87±10,66	21,99±31,57	5,54±30,47

BPI'15 žurnalo prognozavimas buvo mažiausiai tikslus – teisingai prognozuota tik 53,01±31,54 % visų įvykių ir teisingai intervale prognozuota 21,36 % visų įvykių. Taip pat šio žurnalo atveju neteisingai prognozuota vidutiniškai 21,36±23,27 % įvykių. Mažas procentinis teisingų spėjimų kiekis gali būti paaiškinamas tuo, kad žurnale iš viso yra 289 unikalūs galimi įvykiai ir tik 1 196 egzemplioriai. Taip pat, žurnale esantys įvykiai turi mažai duomenų atributų, o tai nulemia ir prastesnį tikimybės išvedimo rezultatą. Apibendrinant galima teigti, kad žurnale esantys duomenys yra sudėtingo proceso vykdymo istorija ir žurnale yra per mažai duomenų tam, kad būtų sudarytas tikslus BBN modelis. BPI'12 prognozavimo rezultatai yra daug geresni - teisingai prognozuota 73,49±31,36 % įvykių, o teisingai intervale – 3,61±11,31 % įvykių. Visgi, neteisingai prognozuota vidutiniškai 13,7±18,86 % įvykių ir iš viso praleista vidutiniškai 9,2±25,04 % įvykių. Galiausiai, EIMSD žurnale esantys duomenys yra dinamiškiausio proceso, kurio kontrolės srauto valdymas yra neapibrėžtas. Nepaisant to, pasiūlytas metodas teisingai prognozavo vidutiniškai 67,6±32,54 % įvykių, o teisingai intervale – 21,36 % įvykių. EIMSD atveju neteisingai prognozuota vidutiniškai 21,99 % įvykių, tačiau dalis įvykių yra visiškai nenuspėjami ir priklauso nuo išorinių veiksnių, kurie žurnale neišreikšti. Apibendrinant visus prognozavimo rezultatus, galima teigti, kad visiems žurnalams metodas vidutiniškai praleidžia 5,54–9,2 % įvykių. Būtina atlikti papildomus tyrimus, siekiant išsiaiškinti priežastis, lemiančias stabilų praleidžiamų įvykių kiekį, bei būdus, kaip galima sumažinti šį kiekį.

Paskutinis eksperimentinis tyrimas vertina metodo VP imitaciją. VP imitacija buvo atlikta remiantis eksperimento aprašu – imituoti iš pradinio žurnalo sukurti imitaciniai modeliai bei sugeneruoti imitacijos rezultato įvykių žurnalai. Po to, šie įvykių žurnalai buvo tarpusavyje palyginti. Gauti rezultatai pateikti S3.4 lentelėje.

S3.4 lentelė. Imitacijos vertinimo rezultatai skirtingiems žurnalams

Žurnalo pavadinimas	SEL atitiktis GMM		GEL atitiktis SMM	
	Egzemplioriaus atitiktis	Atkartojimo atitiktis	Egzemplioriaus atitiktis	Atkartojimo atitiktis
Synthetic log	0,795	1,000	1,000	1,000
BPI12	0,821	0,817	0,814	0,771
BPI13	0,928	0,906	0,438	0,582
BPI15	0,572	0,585	0,886	0,832
EIMSD	0,982	0,995	0,979	0,987

SEL – pradinis įvykių žurnalas, GEL – imitacijos rezultato įvykių žurnalas, GMM – Petri tinklas išgautas iš GEL, SMM – Petri tinklas išgautas iš SEL.

SEL atkartojimas GMM atitinka mažiau, nei GEL atkarojimas SMM. Tai gali būti paaiškinama tuo, kad GEL yra mažiau egzempliorių, todėl ir išgautas GMM yra mažiau atitinkantis, o SEL esantys duomenys atspindi sudėtingesnę elgseną, kurios GMM neleidžia. Tai įrodoma tuo, kad Synthetic žurnalo procesas gan paprastas, o mažas egzempliorių skaičius atspindi visą proceso elgseną – Synthetic žurnalas atitiko GMM (0,929) panašiai kaip ir GEL atitiko SMM (0,941). Bendru atveju, imituoto proceso atitiktis pradinio žurnalo procesui buvo aukštesnė nei 0,814. Šis teiginys negalioja automobilių gamybos proceso žurnalo atveju, nes žurnale yra mažai duomenų atributų ir todėl metodas nesugeba tiksliai imituoti proceso srauto.

Iš gautų rezultatų matyti, kad SEL atitiktis GMM varijuoja labiau nei GEL atitiktis SMM. Tai gali būti paaiškinama tuo, kad SEL yra didesnis istorinio VP vykdymo duomenų rinkinys – pavyzdžiui, BPI'15 žurnalo atveju, pradinis žurnalas turi 289 unikalius įvykius ir 1 159 egzempliorius, o sugeneruotame žurnale buvo tik 250 egzempliorių ir 136 unikalūs įvykiai. Tai lemia, kad sugeneruotas įvykių žurnalas silpniau išreiškia proceso elgseną, nei pradinis įvykių žurnalas.

Norint įvertinti, kokią įtaką imituotų egzempliorių skaičius žurnale daro atitikčiai tarp pradinio žurnalo ir imitacijos rezultato, buvo atliktas papildomas eksperimentas. Eksperimento metu imituotas skirtingas egzempliorių skaičius (100–1 000) ir vertinta kaip kinta sugeneruoto įvykių žurnalo atitiktis pradiniam įvykių žurnalui. Eksperimentams pasirinktas BPI'12 žurnalas. Gauti rezultatai pateikti S3.5 lentelėje.

S3.5 lentelė. Imitacijos vertinimo rezultatai skirtingiems egzempliorių kiekiams

Egzempliorių kiekis, vnt.	SEL atitiktis GMM		GEL atitiktis SMM	
	Egzemplioriaus atitiktis	Atkartojimo atitiktis	Egzemplioriaus atitiktis	Atkartojimo atitiktis
100	0,593	0,572	0,913	0,885
200	0,649	0,627	0,839	0,848
300	0,699	0,653	0,810	0,766
400	0,752	0,732	0,819	0,833
500	0,821	0,817	0,814	0,710
600	0,821	0,817	0,817	0,775
700	0,840	0,831	0,821	0,778
800	0,835	0,833	0,825	0,839
900	0,800	0,773	0,830	0,790
1 000	0,830	0,88	0,820	0,778

GEL atitiktis SMM mažėja, didėjant imituotų egzempliorių kiekiui, o SEL atitiktis GMM mažėja. mažėjant imituotų egzempliorių skaičiui. Abiem atvejais, atitiktis stabilizuojasi ties 500 egzempliorių. Tai paaiškina kodėl BPI'15 žurnalo atitikties rezultatai buvo tokie netikslūs – procesas yra pakankamai sudėtingas (iš viso 289 unikalūs įvykiai), imituota tik 250 egzempliorių, ir to nepakako atitikties stabilizavimui. Tai taip pat įrodo GEL ir SEL unikalių įvykių skaičiaus palyginimas (139 galimi unikalūs įvykiai GEL ir 289 unikalūs įvykiai SEL).

Bendrosios išvados

1. Procesų gavybos metodų analizė atskleidė, kad egzistuojantys metodai yra specializuoti ir šioje tyrimų kryptyje trūksta bendrinių analizės metodų. Prognozavimui skirti metodai specializuojasi atskirų duomenų prognozei, pvz., trukmei, ir negali prognozuoti dalykinės srities duomenų.
2. Verslo procesų imitacijos metodų analizė parodė, kad, nors imitacija ir yra vertinga VP analizei, tačiau egzistuoja daug trūkumų – sukurtas tik vienas verslo procesų imitacijos modelių standartas, egzistuojantys įrankiai naudoja individualias modeliavimo kalbas, modelių kūrimas reikalauja specializuotų įgūdžių, todėl imitacinių modelių kūrimas reikalauja daug rankinio darbo.
3. Pasiūlytas metodas, skirtas išgauti Bajeso tikimybinius modelius (BBN) iš įvykių žurnalų, sukuria tikimybinį verslo modelį, todėl įgalina sprendimų paramą, suteikdamas galimybę išvesti dabar vykdomų verslo procesų egzempliorių tikimybes bei prognozuoti tolimesnį VP egzempliorių vykdymą.
4. Pasiūlytas transformacijos iš BBN į DPBS modelius metodas sumažina rankinį darbą, reikalingą pradinių imitacinių modelių kūrimui, ir sukurtų imitacinių modelių vykdymas atspindi pradiniuose žurnaluose esančius istorinius VP vykdymo duomenis 57,2–92,8 % tikslumu naudotiems įvykių žurnalams..
5. Pasiūlytas būdas BBN išgavimui ir imitacinių modelių kūrimui leidžia atlikti bendrosios VP elgsenos analizę ir analizuoti vykdomus VP egzempliorius bei prognozuoti tolimesnį jų vykdymą.
6. Eksperimentiniai pasiūlytų metodų tyrimai parodė, kad:
   a) pasiūlytas metodas gali išgauti beciklius kryptinius grafus;
   b) vykdomų egzempliorių tikimybinio vertinimo tikslumas siekia 63–98 %;
   c) vykdomo VP egzemplioriaus būsimas įvykis tinkamai prognozuojamas 72–87 % atvejų;
   d) sukurti imitaciniai modeliai pasiekia 58–98 % atitiktį tarp pradinio įvykių žurnalo bei imitacijos rezultato.

Annexes[1]

Annex A. Author's Declaration of Academic Integrity

Annex B. The Co-Authors' Agreements to Present Publications Material in the dissertation

Annex C. Copies of Scientific Publications by the Author on the Topic of the Dissertation

[1] The annexes are supplied in the enclosed compact disc.

Titas SAVICKAS

RESEARCH ON BUSINESS PROCESS PREDICTION AND SIMULATION USING EVENT LOG ANALYSIS METHODS

Doctoral Dissertation

Technological Sciences,
Informatics Engineering (07T)

Titas SAVICKAS

VERSLO PROCESŲ PROGNOZAVIMO IR IMITAVIMO TAIKANT SISTEMINIŲ ĮVYKIŲ ŽURNALŲ ANALIZĖS METODUS TYRIMAS

Daktaro disertacija

Technologijos mokslai,
Informatikos inžinerija (07T)

2017 11 17. 10,0 sp. l. Tiražas 20 egz.
Vilniaus Gedimino technikos universiteto
leidykla „Technika“,
Saulėtekio al. 11, 10223 Vilnius,
http://leidykla.vgtu.lt
Spausdino BĮ UAB „Baltijos kopija“,
Kareivių g. 13B, 09109 Vilnius