DISCOVERY OF DECISION MODELS COMPLEMENTARY TO PROCESS MODELS

EKATERINA BAZHENOVA

BUSINESS PROCESS TECHNOLOGY GROUP
HASSO PLATTNER INSTITUTE, UNIVERSITY OF POTSDAM

DISSERTATION
ZUR ERLANGUNG DES AKADEMISCHEN GRADES
"DOCTOR RERUM NATURALIUM"
– DR. RER. NAT. –

February 2018

Ekaterina Bazhenova: *Discovery of Decision Models Complementary to Process Models,* eingereicht an der Digital-Engineering-Fakultät des Hasso-Plattner-Instituts und der Universität Potsdam. February 2018

Published online at the
Institutional Repository of the University of Potsdam:
URN urn:nbn:de:kobv:517-opus4-410020
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus4-410020

ABSTRACT

Business process management is an acknowledged asset for running an organization in a productive and sustainable way. One of the most important aspects of business process management, occurring on a daily basis at all levels, is decision making. In recent years, a number of decision management frameworks have appeared in addition to existing business process management systems. More recently, Decision Model and Notation (DMN) was developed by the OMG consortium with the aim of complementing the widely used Business Process Model and Notation (BPMN). One of the reasons for the emergence of DMN is the increasing interest in the evolving paradigm known as the separation of concerns. This paradigm states that modeling decisions complementary to processes reduces process complexity by externalizing decision logic from process models and importing it into a dedicated decision model. Such an approach increases the agility of model design and execution. This provides organizations with the flexibility to adapt to the ever increasing rapid and dynamic changes in the business ecosystem. The research gap, identified by us, is that the separation of concerns, recommended by DMN, prescribes the externalization of the decision logic of process models in one or more separate decision models, but it does not specify *how* this can be achieved.

The goal of this thesis is to overcome the presented gap by developing a framework for discovering decision models in a semi-automated way from information about existing process decision making. Thus, in this thesis we develop methodologies to extract decision models from: (1) control flow and data of process models that exist in enterprises; and (2) from event logs recorded by enterprise information systems, encapsulating day-to-day operations. Furthermore, we provide an extension of the methodologies to discover decision models from event logs enriched with fuzziness, a tool dealing with partial knowledge of the process execution information. All the proposed techniques are implemented and evaluated in case studies using real-life and synthetic process models and event logs. The evaluation of these case studies shows that the proposed methodologies provide valid and accurate output decision models that can serve as blueprints for executing decisions complementary to process models. Thus, these methodologies have applicability in the real world and they can be used, for example, for compliance checks, among other uses, which could improve the organization's decision making and hence it's overall performance.

ZUSAMMENFASSUNG

Geschäftsprozessmanagement ist eine anerkannte Strategie, um Unternehmen produktiv und nachhaltig zu führen. Einer der wichtigsten Faktoren des Geschäftsprozessmanagements ist die Entscheidungsfindung – tagtäglich und auf allen Ebenen. In den letzten Jahren wurden – zusätzlich zu existierenden Geschäftsprozessmanagementsystemen – eine Reihe von Frameworks zum Entscheidungsmanagement entwickelt. Um die weit verbreitete Business Process Model and Notation (BPMN) zu ergänzen, hat das OMG-Konsortium kürzlich die Decision Model and Notation (DMN) entwickelt. Einer der Treiber für die Entwicklung der DMN ist das wachsende Interesse an dem aufstrebenden Paradigma der "Separation of Concerns" (Trennung der Sichtweisen). Dieses Prinzip besagt, dass die Prozesskomplexität reduziert wird, wenn Entscheidungen komplementär zu den Prozessen modelliert werden, indem die Entscheidungslogik von Prozessmodellen entkoppelt und in ein dediziertes Entscheidungsmodel aufgenommen wird. Solch ein Ansatz erhöht die Agilität von Modelentwurf und –ausführung und bietet Unternehmen so die Flexibilität, auf die stetig zunehmenden, rasanten Veränderungen in der Unternehmenswelt zu reagieren. Während die DMN die Trennung der Belange empfiehlt und die Entkopplung der Entscheidungslogik von den Prozessmodellen vorschreibt, gibt es bisher keine Spezifikation, wie dies erreicht werden kann. Diese Forschungslücke ist der Ausgangspunkt der vorliegenden Arbeit.

Das Ziel dieser Doktorarbeit ist es, die beschriebene Lücke zu füllen und ein Framework zur halbautomatischen Konstruktion von Entscheidungsmodellen zu entwickeln, basierend auf Informationen über existierende Prozessentscheidungsfindung. In dieser Arbeit werden die entwickelten Methoden zur Entkopplung von Entscheidungsmodellen dargestellt. Die Extraktion der Modelle basiert auf folgenden Eingaben: (1) Kontrollfluss und Daten aus Prozessmodellen, die in Unternehmen existieren; und (2) von Unternehmensinformationssystemen aufgezeichnete Ereignisprotokolle der Tagesgeschäfte. Außerdem stellen wir eine Erweiterung der Methode vor, die es ermöglicht, auch in von Unschärfe geprägten Ereignisprotokollen Entscheidungsmodelle zu entdecken. Hier wird mit Teilwissen über die Prozessausführung gearbeitet. Alle vorgestellten Techniken wurden implementiert und in Fallstudien evaluiert – basierend auf realen und künstlichen Prozessmodellen, sowie auf Ereignisprotokollen. Die Evaluierung der Fallstudien zeigt, dass die vorgeschlagenen Methoden valide und akkurate Entscheidungsmodelle produzieren, die als Blau-

pause für das Vollziehen von Entscheidungen dienen können und die Prozessmodelle ergänzen. Demnach sind die vorgestellten Methoden in der realen Welt anwendbar und können beispielsweise für Übereinstimmungskontrollen genutzt werden, was wiederum die Entscheidungsfindung in Unternehmen und somit deren Gesamtleistung verbessern kann.

PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

- Ekaterina Bazhenova, Stephan Haarmann, Sven Ihde, Andreas Solti, and Mathias Weske. *Discovery of Fuzzy DMN Decision Models from Event Logs*. In *29th International Conference on Advanced Information Systems Engineering*, volume 10253 of *Lecture Notes in Computer Science*, pages 629–647. Springer International Publishing, 2017.

- Ekaterina Bazhenova, Francesca Zerbato, and Mathias Weske. *Data-Centric Extraction of DMN Decision Models from BPMN Process Models.* In *5th International Workshop on Declarative / Decision / Hybrid Mining & Modeling for Business Processes*, volume 308 of Lecture Notes in Business Information Processing, pages 542–555. Springer International Publishing, 2018.

- Ekaterina Bazhenova, and Mathias Weske. *Optimal Acquisition of Input Data for Decision Taking in Business Processes*. In *32nd ACM Symposium on Applied Computing*, pages 703–710. ACM Digital Library, 2016.

- Ekaterina Bazhenova, Susanne Buelow, and Mathias Weske. *Discovering Decision Models from Event Logs*. In *19th International Conference on Business Information Systems*, volume 255 of *Lecture Notes in Business Information Processing*, pages 237–251. Springer International Publishing, 2016.

- Laurent Janssens, Ekaterina Bazhenova, Johannes De Smedt, Jan Vanthienen, and Marc Denecker. *Consistent Integration of Decision (DMN) and Process (BPMN) Models*. In *28th International Conference on Advanced Information Systems Engineering – Forum/Doctoral Consortium*, volume 1612 of *CEUR Workshop Proceedings*, pages 121–128. CEUR-WS.org, 2016.

- Ekaterina Bazhenova, and Mathias Weske. *Deriving Decision Models from Process Models by Enhanced Decision Mining*. In *3th International Workshop on Decision Mining & Modeling for Business Processes*, volume 256 of *Lecture Notes in Business Information Processing*, pages 444–457. Springer International Publishing, 2016.

- Kimon Batoulis, Andreas Meyer, Ekaterina Bazhenova, Gero Decker, and Mathias Weske. *Extracting Decision Logic from Process Models*. In *27th International Conference on Advanced Information*

Systems Engineering, volume 9097 of *Lecture Notes in Computer Science*, pages 349–366. Springer Berlin Heidelberg, 2015

- Ekaterina Bazhenova, and Mathias Weske. *A Data-Centric Approach for Business Process Improvement Based on Decision Theory.* In *15th International Conference on Business Process Modeling, Development and Support*, volume 175 of *Lecture Notes in Business Information Processing*, pages 242–256. Springer Berlin Heidelberg, 2014.
- Ekaterina Bazhenova. *Support of Decision Tasks in Business Process Management*. In *Central European Workshop on Services and their Composition*, volume 1140 of *CEUR Workshop Proceedings*, pages 21–26. CEUR-WS.org, 2014.

In addition to above publications, I was also involved in the following research indirectly contributing to this thesis:

- Luise Pufahl, Ekaterina Bazhenova, and Mathias Weske. *Evaluating the Performance of a Batch Activity in Process Models.* In *10th International Workshop on Business Process Intelligence*, volume 202 of *Lecture Notes in Business Information Processing*, pages 277–290. Springer Berlin Heidelberg, 2015.

ACKNOWLEDGMENTS

Upon the closing of such a big and intense period of my life, dedicated to the creation of this thesis, I am looking back and I see a lot of lights shining in the windows, that opened to support me throughout this journey.

Foremost, my journey would not have started, lasted, and finished if not for the support from my supervisor Prof. Dr. Mathias Weske. Thank you for providing me with the opportunity to write the thesis at the BPT group, and for believing in me, especially during the strenuous process of finding a PhD research topic. I am very grateful for all your suggestions and tips which guided me throughout the ups and downs of my PhD, and which in the end led me to the completion of my thesis.

I would also like to thank Prof. Dr. Jan Vanthienen and Prof. Dr. Ir. Hajo A. Reijers for agreeing to review this thesis, and taking their time and effort in this process.

Starting from the second year of my PhD study, I became a member of the HPI Research School, which provided me with the opportunities to enhance my academic skills through a series of seminars, workshops, and retreats. I enjoyed very much the numerous and diverse discussions with the other Research School members. I am grateful for the generous Research School support, which contributed towards this journey and allowed me to attend many international scientific events all over the world.

During my PhD lifetime, I was very lucky to be able to go on a research visit to the KU Leuven and collaborate with Prof. Dr. Jan Vanthienen and the other colleagues from the Leuven Institute for Research on Information Systems. Through the many discussions with them, I learned a lot about integrated business processes and decision management, which helped me further with the creation of my thesis. I really appreciate this experience and big thanks to Jan for supporting this collaboration.

The BPT group members also contributed immensely to my professional and personal life. A very special thanks goes to Luise, Andreas M., and Matthias, who were patiently helping me with all my endless questions throughout the PhD lifetime. I am also very grateful to Andreas S., Marcin, Thomas, Rami, Anne, and Nico in supporting me in the process of searching for a PhD research topic by sharing their experience and ideas. A big thank you also goes out to my more recent colleagues Kimon, Adriatik, Sankalita, Alaaeddine, and Francesca, who have helped to shape and finalize my thesis through multiple discussions. My special thanks goes to Alaaeddine, Andreas

M., Sankalita, Francesca, and Adriatik for proofreading parts of this thesis. Additionally, I would like to thank all BPT members and students who became co-authors of the publications which contributed to this thesis.

I gratefully acknowledge Dr. Sharon Nemeth, a translator at HPI, for proofreading parts of this thesis with respect to the English language. She provided me with valuable tips attaining the scientific usage of the English language.

I am also very grateful to Katrin Heinrich and Sabine Wagner for their invaluable help with the organizational and document related matters, which they invariably provided to me throughout all these years.

My PhD lifetime would be very miserable without my friends who supported me throughout the whole journey. My great thanks goes to all my old friends, new friends and particularly those who helped me navigate life in Germany – you know who you are.

It goes without saying that I would like to specially thank my father, mother, sister, and brother for their encouragement and support. Thank you for making me who I am today, for believing in me, and making my life fulfilled with love and support. Your teachings have given me the tools that I needed to complete my thesis and I am very grateful to you for it.

Ultimately, I am very grateful to my love Donovan who lit the brightest and warmest light on my whole life. I know it is not easy to be a second half of a PhD candidate, but you have succeeded! Big thanks for contributing to the development of my research strategies, for our endless discussions, and the enduring proofreading of my works. I am forever thankful for your unconditional support in all life situations, for accepting me for who I am, and for accompanying me during this journey with patience, humor, and love.

My very last acknowledgment goes to Anton Chekhov, who inspires me with his philosophy: ”If there is a meaning and a purpose in life, that meaning and purpose is not our happiness, but something greater and more rational. Do good!” I agree with it, and I believe in the special role of science in making the world a better place. There are so many ways to build a better world, and every single person can contribute towards it. Therefore, I want to dedicate this thesis to all those who are, were, and will be contributing towards promoting positive changes in the world, to create a better environment for humanity.

Information is a source of learning. But unless it is organized, processed, and available to the right people in a format for decision making, it is a burden, not a benefit.
— William Pollard

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I

BACKGROUND

1 INTRODUCTION

In 2016, the European Parliament adopted a set of regulations for the collection and use of personal information. Scheduled to take effect as a law across the European Union in 2018, the law foresees a "right to explanation," whereby users can ask for clarification of an algorithmic decision affecting them. In order to ensure compliance, many companies are re-examining their processes [5].

Providing integrated business process and decision management frameworks supports businesses against the newly posed challenges. It does this by providing tools to explicitly model and execute processes and decisions. The lifecycle of integrated business process and decision management starts with the design phase, in which processes and decisions are identified and modeled. Existing frameworks rely heavily on experts modeling idealized "to-be" processes and decisions [183]. To assist companies with the automated design of real process decisions, knowledge about "as-is" process decision making needs to be retrieved.

Often in practice, decision logic is either explicitly encoded in process models through control flow structures [18], or it is implicitly contained in process execution logs [23]. According to the separation of concerns principle, modeling decisions complementary to processes is the recommended approach [142]. To this end, our work proposes a set of methodologies for the derivation of decision models from both process models and event logs. The derived decision model explains "as-is" decision making, and serves as a blueprint to execute decisions complementary to the process model. The presented thesis is a detailed report on the conducted research and the results achieved from our experiments.

This chapter introduces the thesis and outlines its content. In Section 1.1 we investigate the drivers for discovery of decision models complementary to process models. Based on this setting, we derive the problem statement in Section 1.2. The main contributions of the thesis are discussed further in Section 1.3. Section 1.4 presents our research methodology, which is used as a foundation for conducting this research, and explains the thesis structure.

1.1 MOTIVATION

Competing in the Information Age, companies strive to run their business processes in a productive and sustainable way. It has been widely acknowledged that business process management (BPM) offers competitive advantages to run companies efficiently [16, 63, 202]. Decision making plays a crucial role in business process management. Integrated business process and decision management improves the process by both focusing on the way decisions are made and directing the processes that carry out the decisions [144, 177, 194, 197].

An interest towards decision support in business processes is indicated by a number of both theoretical works on decision ontologies and notations [30, 104, 210], and industrial decision frameworks complementary to business process management systems [6, 48, 168, 172]. Recently, the Decision Model and Notation (DMN) standard was developed by the OMG consortium, whose Board of Directors consists of representatives of such companies as SAP, IBM, Microsoft, HP, Oracle, and other major firms [136]. This trend indicates a strong interest of major market players in an interoperable standard for business decision management.

One of the primary prerequisites for emergence of the DMN standard is the paradigm of separation of process and decision concerns, which has gained in importance over the last years [60, 94, 177, 178, 197]. The paradigm states that modeling decisions separated from process models can reduce process complexity by excluding decision logic from process models and imported into a dedicated decision model. This allows reusage of the same decision model in different processes. Such a modeling approach also increases agility of processes, since decisions typically have a more dynamic nature in comparison to processes. According to [178], making changes in decisions that are complementary to processes reduces time and costs, provides a more rapid response to business risks, and results in fewer missed opportunities. As an example, the authors use a process of the US government agency on managing vehicle licensing payment. The licensing management, at across more than 150 channels, involves approximately 50 million registered vehicles. Separating the fee calculation decision from the rest of the process enabled an easy updating of thousands of rules by non-technical users. As a result, around 13,000 hours of maintenance work were saved in one year.

In this scenario, there arise a lot of issues during integrated modeling and executing of processes and decisions. The lifecycle of integrated business processes and decision management starts with the design phase, in which business processes and decisions need to be identified and modeled. Through the use of support tools [6, 48, 168, 172], modeling languages such as Business Process Model and Notation (BPMN) [138] or DMN are well suited for both process experts

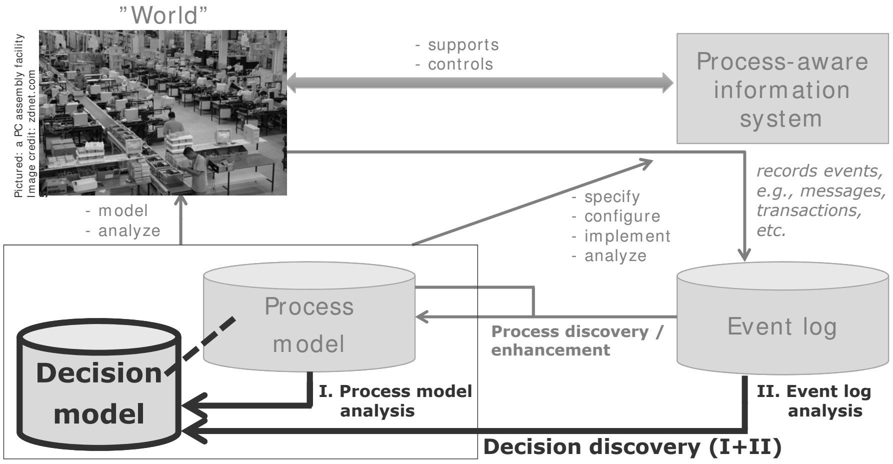

"World"
Pictured: a PC assembly facility
Image credit: zdnet.com
- supports
- controls
Process-aware information system
records events, e.g., messages, transactions, etc.
- model
- analyze
- specify
- configure
- implement
- analyze
Process model
Event log
Process discovery / enhancement
Decision model
I. Process model analysis
II. Event log analysis
Decision discovery (I+II)

Figure 1: Two perspectives of discovering decision models complementary to process models (in bold) covered in this thesis, with respect to the process mining picture adapted from [181, 184]. Information systems record event logs that capture information about the execution of real-world processes. Process models can be either modeled by experts or discovered from event logs with the help of the existing process mining techniques. Both process models and event logs can serve as a source of information about existing decision making at an enterprise.

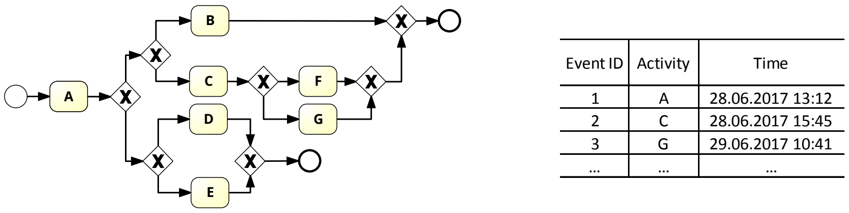
and end users. State-of-the-art decision management tools support expert's design of process decisions, often with the goal of further automation of decision making. However, there exist situations where modeling of decisions by experts needs to be additionally supported, e.g., in the situations where companies run a lot of processes involving decision making. In our work, we want to assist companies with the design of explanatory decision models relying on information about process and its execution. In such a way, we propose to use the approach, where decision models are designed based on knowledge about "as-is" process decision making.

Despite extensive research on how to effectively design process models (e.g., [63, 156, 183, 202]) and decision models (e.g., [160, 177, 178, 208]) independently, the question of the integrated design of process and decision models has received far less attention. While the separation of concerns paradigm, recommended by DMN, *prescribes* externalizing the decision logic of process models in one or more separate decision models, it does not specify *how* this should be done. That said, the integrated usage of business process and decision management approaches offers new opportunities for discovering decisions from information about existing process decision making. Thus, there is a need for a comprehensive methodology supporting companies in designing decision models that complement process models.

As prescribed by the OMG consortium, in designing decision models complementary to process models [142], decision discovery is closely related to process discovery. The main goal of process mining is to provide the automated discovery of process-related information from event logs created by IT systems. In order to position our work, in Figure 1, adapted from Van der Aalst et al. [181, 184], we present an overview of the general process mining setting. The figure shows perspectives of the discovery of decisions complementary to processes as an augmentation in bold.

As can be seen in Figure 1, processes are carried out by machines and people with the help of software systems. These software systems record event logs, which range from simple text files to data hidden in complex structures of large databases. Once we have extracted an event log from the IT systems, process mining provides the discovery or enhancement of existing processes, e.g., through usage of *alpha*-algorithm and its extensions [57, 113, 186], heuristic approaches such as genetic algorithms [32, 55], as well as region-based mining algorithms [187], or inductive mining algorithms [108].

Different ways of capturing process decision making exist in a company. These depend on the company's BPM maturity, information systems, expertise and data. An analysis of existing works and interviews with our industrial partners allow us to identify two fundamental approaches for capturing operational decisions: (1) in process models, and (2) in process event logs. Corresponding approaches that serve towards the discovery of decision models complementary to process model are *Process model analysis* and *Event log analysis*. This connection is indicated by bold arrows in Figure 1.



Event ID	Activity	Time
1	A	28.06.2017 13:12
2	C	28.06.2017 15:45
3	G	29.06.2017 10:41
...	...	...

(a) BPMN process model explicitly encoding decision logic in control flow

(b) Event log implicitly storing decision results

Figure 2: Different ways of capturing decision-related process information

1. PROCESS MODEL ANALYSIS. In cooperation with our industrial partners, we analyzed about 1,000 real process models from insurance, banking, and health care domains [18]. Our analysis shows that the decision logic is often encoded in complicated control flow structures, consisting of sequences of exclusive gateways. An abstract BPMN process model from Figure 2a is an example of control flow misuse for modeling decisions, which is hard to read and maintain. Understanding the patterns of incorporating decision logic into control flow assists with the extraction of decision models complementary to pro-

cess models. In this way, decision logic can be externalized into a dedicated decision model, e.g., a DMN decision table which is a tailored compact method to describe the decision logic [142]. Thereby, the corresponding process control flow can be significantly simplified.

Additionally, the decision logic can be hidden in the process model in the form of implicit data dependencies [19, 20, 21, 23]. Understanding the patterns of incorporating decision logic into process data objects can be useful for extracting decision models from process models. In this thesis, we propose the methodologies to discover decision models complementary to process models from both the control and the data flow of the process model.

II. EVENT LOG ANALYSIS. Process decision making can be captured and analyzed through an event log (see Figure 2b corresponding to the process model from Figure 2a). For instance, the event log can contain claim data and logged expert decisions for the credit-risk assessment process. In such cases, the decisions made by experts are only implicitly contained in the event log, e.g., in the event log in Figure 2b it cannot be seen *why* activity *C*, and not some other activity, was executed after activity *A*. In such cases, the decision logic can be derived by solving a classification problem, where the classes are possible outcomes of process decisions, and the training examples are the process instances recorded in the event log. A lot of works exist that use statistical means for the discovery of decision rules [205], but their direct application can lead to the unjust treatment of an individual applicant. Thus, for companies it is important to derive interpretive decision models from event logs explaining *how* decisions were taken.

Decisions are often represented by rules based on Boolean algebra, e.g., “If client age is more than 45 years old, then the discount is 25%.” The formal nature of such decisions is often hard for interpretation and utilization in practice. This is because imprecision is intrinsic to real-life decisions. Operations research considers fuzzy logic, based on fuzzy algebra, as a tool dealing with partial knowledge [89, 193]. Fuzzy rules represent strings encoding the semantic meaning of a certain probability behind a value range [209], e.g., “If loan duration is long, then risk is very high.” Since meaning can be derived directly from the representation, it is generally considered that fuzzy rules are more comprehensible compared to crisp rules [89]. Moreover, using literals across rules increases decision flexibility, as it allows a consistent adaptation of all rules by adjusting only the underlying mappings. In this thesis, we explore the possibility of incorporating fuzziness into decision models based on the example of DMN. Consequently, we provide methodologies for discovering both classic and fuzzy decision models from event logs.

To sum it up, we utilize the decision-related information with respect to two perspectives described above, and propose methodologies for deriving decision models from both process models and their corresponding execution logs. The constructed decision models explain existing process decision making and serve as blueprints for executing decisions complementary to process models. This has been our core motivation for commencing the research presented in this thesis.

1.2 PROBLEM STATEMENT

The focus of this research is to provide background on the paradigm of the separation of process and decision concerns, different ways of capturing decision logic in processes, and externalizing process decision logic in dedicated models complementary to input process models. Our focus group is composed of business process and decision management stakeholders, such as process and decision officers, owners, designers, participants, knowledge workers, system architects, and developers. The broader group of our goal stakeholders encompasses all kind of users, vendors, and also academic circles related to enterprise information systems. As we aim at designing explanatory decision models that can be executed complementary to process, those stakeholders who run processes involving frequent decision-making regulated by strict guidelines benefit the most. An example industry is the banking domain managing such processes as an evaluation of customer creditworthiness, claim acceptance, eligibility decision in social security, etc. [208]. Another example domain is healthcare, which includes such processes as determining the appropriate type of check to be performed on patients, assigning them to nurses, or determining patient-oriented care programs [132].

The goal of this research is to provide stakeholders with a comprehensive methodology on designing decision models of their process decision making complementary to process models. Taking into account that information about operational decisions can be derived from knowledge about current processes of an enterprise (see Section 1.1), we formulate the research objective of this thesis as follows.

> DESIGN AND EVALUATE A FORMAL FRAMEWORK TO DISCOVER DECISION MODELS FROM PROCESS MODELS AND EVENT LOGS.

Further, we determine a set of goals that shall be met in the course of this thesis in order to achieve our research objective.

1. *Provide a specification of decision models that can effectively be used to design and execute decision models complementary to process models.*
   In order to address our research objective, an understanding is necessary of the requirements for decision models and their desired properties. To achieve this, it first has to be determined what types of decisions are suitable for business process management, and what properties of decision models designed complementary to process models are relevant. Further, an analysis of existing practices and tools suitable for modeling process decisions should be done to identify the stakeholders experience on process decision design. The best practices and identified gaps in modeling decisions complementary to processes form a requirements base for the output decision model.

2. *Identify constructs of state-of-the-art encapsulation of decision logic in process models and their execution logs.*
   The "process-first" nature of the approach that we pursue in our thesis infers that we want to provide support for designing decision models based on knowledge about existing process decision making at enterprises. In order to implement this, a rigorous analysis of possible ways and practices of representing decisions in processes should be devised. Moreover, most business processes today are not designed to be decision aware [197] (this served as a premise for creating the DMN standard complementary to the BPMN standard). Lacking proper integration with process models, the decision making has been implemented in multiple ways, e.g., by experts or rule engines working externally from process engines. In such cases, decision logic is often implicitly encoded in event logs of information systems, such that process execution data is another source of valuable knowledge about process decision making. Our goal is to identify and summarize such ways of representing and recording knowledge about process decision making which can be used for extracting information relevant for decision models.

3. *Design methodologies for extraction of decision models from process models and event logs.*
   To ensure that stakeholders can effectively utilize the concept of designing decision models complementary to process models, we want to support the process of discovering decision models from process models and event logs with corresponding methodologies. The methodologies should take as input process models or event logs, and provide a step-by-step approach for obtaining output decision models that can be utilized complementary to process models. In case only event logs are available, the discovery of process models can be done by process mining

techniques. These are well explored in literature [184], and are beyond the scope of this thesis. As we expect our techniques on the discovery of decision models from process knowledge to be used by a broad group of stakeholders, our methods shall be represented by comprehensible step-by-step methodologies, and be demonstrated based on real-world examples.

4. *Evaluate how effectively the developed methodologies support the discovery of decision models from process models and event logs.*
   The applicability and usefulness of the proposed methodologies for the discovery of decision models shall be assessed through comprehensive evaluation. We need to investigate how proposed methodologies can be implemented and observe how they interact with a real-world context. The proposed methodologies should require minimized effort for stakeholders in extracting decision models from process models and event logs, and achieving distinct separation of process and decision concerns in output models.

The presented goals define the direction of the research conducted in this thesis. We develop the thesis structure with respect to these goals further in Section 1.4.

1.3 CONTRIBUTIONS

As indicated by the bold arrows in Figure 1, we derive decision models either from process model or event logs, depending on what is available at an enterprise. Therefore, our methodologies on deriving decision models from process models (see Chapters 4, 5) and from event logs (Chapters 6, 7) are independent, though they are not meant to be exclusive, but rather complementary. If no process model exists, numerous process mining techniques mentioned above can be used to discover process models from event logs. At the same time, the event logs might contain additional information related to process decisions which is not part of process models. Therefore, if only an event log is available as an input, discovery of a decision model from this event log cannot be replaced through the consequent discovery of a process model and further discovery of a decision model from the derived process model.

Thus, to answer the research questions formulated in Section 1.2, this thesis provides a novel approach for discovering decision models complementary to process models that takes process model and execution logs as inputs (see the outline of our approach in bold in Figure 1). Some parts of this thesis have been previously published in [18, 19, 20, 21, 22, 23, 24, 25, 96]. The key contributions of this thesis are:

- *A formal specification of the elements and the relationships that form a decision model which can be designed and executed complementary to process model. (See Goal 1 from Section 1.2.)*
  Modeling languages are typically used by both individuals and organizations, all having diverse backgrounds and stemming from various industries. When developing a methodology for decision design, this diversity needs to be taken into consideration. One way of ensuring this is to develop a methodology that can be used to create decision models according to well-established formalization of entities and relationships that form a decision model which has the following properties: (1) it can be used complementary to process, (2) separation of process and decision logic is supported. Since real-life decision processes are often exposed to imprecision, we extend our base formal concept of decision model with fuzzy logic which was introduced by Zadeh in [209] and has been established as an effective tool for dealing with partial input knowledge for decision making [27].

- *Methodologies to extract decision models from control flow and data of process models. (See Goals 2-3 from Section 1.2.)*
  - *Control flow patterns.* Many real-world process models utilized by stakeholders contain decision logic incorporated into control flow structures of process models, although it is not considered good practice [18]. We utilize such knowledge for our purpose and propose a semi-automatic methodology that enables identification of decision logic in the control flow of process models, derivation of corresponding DMN models, adaptation of the original process model by replacing the decision logic accordingly, and final configuration of the result during post-processing. The identification is pattern-based derived from an intensive analysis of about 1,000 real world process models provided by our project partners.
  - *Data structures.* The decision logic can also be hidden in the process models in the form of implicit data dependencies between data objects [19, 20, 25]. Thus, the decision making can be captured implicitly in attributes of data objects in a process model. For example, the decision activity of choosing a vendor from the list of available ones in a logistic company can be simply captured with an input data object which consists of a list of alternatives and an output data object containing the chosen vendor. Thereby, the decision logic is not represented in the process model. We investigate structures of the incorporation of decision logic into process model data, and we propose a methodology to extract decision models from these data.

- *Methodologies to extract decision models from event logs. (See Goals 2-3 from Section 1.2.)*
    - *Base methodology.* Using information about decisions contained implicitly in the process execution data, we provide a formal framework enabling the extraction of complete decision models from event logs. In particular, we propose a methodology which extends an existing approach [165] to derive control flow decisions from event logs with additional identification of data decisions and dependencies between them. Furthermore, we propose a technique to rebuild decision trees to identify the dependencies between discovered decisions and to overcome the problem of reusing attributes in a dependent decision.
    - *Discovering fuzzy decision models.* To assist companies with the automated construction of fuzzy decision models complementary to process models, we introduce a methodology to discover these models from event logs. Fuzzy decision logic in models is represented by fuzzy decision rules, which we obtain by application of fuzzy learners for process decisions. Hereby, we explore the possibility of applying genetic and NEFCLASS classifiers, and describe needed modifications of the algorithms for the process context. Further, we propose a technique of how to prioritize fuzzy rules during their run-time execution.

Apart from developing the above mentioned methodologies, we evaluate how effectively the introduced methodologies support the discovery of decision models from process model and event logs in Chapter 8. (This corresponds to Goal 4 from Section 1.2.) Our work is aimed to be reusable and extensible with respect to supporting stakeholders with the design of decision models complementary to process models. The introduced specification can be employed by stakeholders to develop their own methodologies for managing decisions complementary to process models alongside the whole lifecycle of integrated process and decision management. The presented methodologies to extract decision models from process models and event logs are open to include additional techniques.

1.4 THESIS STRUCTURE

Our investigation is grounded by a review and analysis of the literature dedicated to existing methods, techniques, and approaches to business process management and decision management. We reinforce our theoretical findings through a series of interviews with practitioners and vendors of business process and decision management systems.

The work in this thesis follows a design science methodology as, for example, suggested by Wieringa [204]. The stakeholders of our study are users and vendors of business process and decision management solutions, represented by both academia and industry. The primary objective of this research is to provide a methodology for extracting decision models taking as inputs (1) process models, and (2) event logs. The reusable artefacts that are the result of this thesis are methodologies to discover decision models from process models and event logs.

This thesis is organized as follows. *Part I* provides the background for the whole thesis. The current *Chapter 1* opens up this part with the motivation, the problem statement, the contributions and the structure of this thesis. *Chapter 2* explores the problem context by providing an overview of the types of decisions maintained in business process management, and it also presents needed definitions from business process and decision management domains. This corresponds to Goal 1 from Section 1.2. *Chapter 3* presents works related to the problem of discovering decision models from event logs. We achieve this by means of a systematic literature review on best practices of process and decision modeling. For the systematic literature review we follow the guidelines proposed in [99].

Part II addresses the presented thesis problem by providing methodologies to discover decision models from decision-related process information. These mentioned methodologies correspond to Goals $2-3$ from Section 1.2. *Chapters 4-5* introduce our methodologies for extraction of decision models from process models. In *Chapter 4*, we propose a methodology to derive decision models from control flow of process models. *Chapter 5* proposes a methodology to derive decision models from process model data. *Chapters 6-7* introduce our methodologies to extract decision models from event logs. *Chapter 6* presents a methodology to derive classic decision models from event logs. *Chapter 7* presents the fuzzy extension of the methodology from *Chapter 6*.

Part III evaluates and discusses the conceptual results presented previously, which corresponds to Goal 4 from Section 1.2. Chapter 8 introduces our evaluation of the presented methodologies. First, we introduce the description of the methods and metrics used in our experiments. Second, we validate our methodologies by implementing corresponding prototypes. Further, we evaluate the presented methodologies by applying them to several real-life and synthetic use cases and discussing the findings. Finally, in Chapter 9, we summarize the thesis results, discuss the limitations of our methodologies, and outline future work directions.

2 PRELIMINARIES

The value of business process management (BPM) has been acknowledged as an essential asset to drive a company [16, 63, 202]. One of the important and challenging BPM aspects is decision making. Exposed to massive volumes of information nowadays, managers are challenged to make highly complex decisions in increasingly shorter time frames. Decision management exists as an independent discipline which has been developing over the last decades [177, 194]. Integrated business process and decision management improve the business activities of a company by both focusing on the way decisions are made and directing the processes that incorporate decision making [144, 177, 194]. Despite extensive research on business process management and decision management as independent disciplines, the question of the integrated usage of these two assets has received far less attention up to now. Designing decision models complementary to process models is a solution bridging the gap between business process and decision modeling. Our work is aimed at supporting this design by providing methodologies for extracting decision models from process models and event logs.

Addressing our goals from Section 1.2, we present formalisms and approaches, which are fundamental for this thesis, in this chapter. In particular, in Section 2.1, we report on the background of business process management and decision management both as independent and as integrated disciplines. Section 2.2 presents an integrated lifecycle for business process and decision management, and discusses each phase of it. Section 2.3 examines the fundamentals of business process and decision modeling. In Section 2.4, we present foundations of process and decision execution. The chapter concludes with a discussion of the goals of the thesis in light of the introduced concepts.

2.1 INTEGRATED BUSINESS PROCESS AND DECISION MANAGEMENT

In this section, we present the background of business process management, followed by the background of decision management. The section concludes with an overview of the state-of-the-art integration of these two disciplines.

2.1.1 *Business Process Management*

The primary purpose of any enterprise is to maximize profits for its owners or stakeholders [71, 72]. To ensure its own competitive performance, a company needs tools to effectively manage its business operations. The widely acknowledged discipline of business process management (BPM) addresses this goal by providing an approach and tools for managing and improving business processes through a holistic process-oriented view [51, 63, 83, 151, 169, 202].

Business processes form the basis of a company's value chain, which represents a set of activities that a company is operating in order to deliver a valuable product or service for the market [151]. On the basis of [202] we define the term business process as follows:

Definition 1 (Business Process). A *business process* consists of a set of coordinated activities that contribute to a specific business goal. A business process is embedded into an organizational, informational and technical environment. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations. ◇

Business process management (BPM) is a branch of knowledge that includes multiple facets, as captured in the definition below, stemming from [184]:

Definition 2 (Business Process Management). *Business process management* is the discipline that combines approaches for the design, execution, control, measurement and optimization of business processes. ◇

Following the development and adoption of information systems, organizations can achieve additional benefits for coordinating the activities involved in business processes. Following [202], [16], and [63], we refer to information systems that exploit an explicit representation of a business process as business process management systems (BPMSes).

Definition 3 (Business Process Management System). A *business process management system* is a generic software system that is driven by explicit process representations to coordinate business process enactment. ◇

BPM aims at achieving both business objectives, such as business performance improvement, and technical objectives, such as the implementation of supporting information systems, e.g., BPMS [159]. In both cases, the BPM objectives need to be aligned with the organization's strategy [162]. In achieving their goals, organizations run BPM projects that represent ongoing tasks of continuous managing and monitoring of their business processes [159].

2.1.2 *Decision Management*

Decision making is one of the most important aspects of business processes in organizations. However, not all decisions are relevant for consideration in business process management. Therefore, we present definitions and properties of enterprise decisions relevant to BPM. These are considered throughout this thesis. Further, we present concepts that are at the core of the decision management discipline.

Which Decisions are Relevant for Business Process Management?

Enterprise decisions vary from routine choices, such as how many items to produce, to unexpected ones such as what to do if a new technology emerges and disrupts the market. According to [16], the decision-making structure in a typical company is similar to a pyramid, the cross section of which is shown in Figure 3. At the top of the decision-making pyramid is the *strategic level*, in which managers develop overall business goals and objectives as part of the enterprise strategic plan. Such decisions are usually rare. They are largely unstructured in the sense that a process prescribing how to take them does not exist. Hereby, these decisions are extremely important for the company, e.g., a company decision to enter a new market over the succeeding years. On the second layer of the decision-making pyramid is the *managerial level*, which covers short- and medium-term plans, budgets, and procedures, e.g., a decision about employee benefits. These decisions are considered as semi-structured. They occur in situations in which established processes help to evaluate potential solutions, but do not provide enough information to lead to a definite recommended decision. The bottom of the decision-making pyramid is the *operational level*, in which employees develop and maintain core activities required to run the daily operations that are affected by short-term business strategies. These decisions are normally structured in the sense that established processes exist prescribing how they should be taken. Operational decisions are made frequently, and they are repetitive in nature, e.g., determining whether a customer is eligible for a benefit.

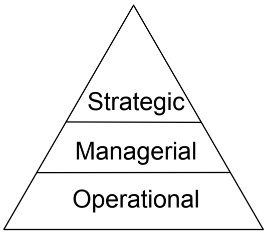


Figure 3: Types of enterprise decisions, adapted from [16]

If we look again at the motivation for the separation of process and decision concerns in Section 1.1, making changes in decisions that are designed complementary to processes potentially reduces the time and costs of process execution. Decisions that incorporate high change components of business processes are considered the operational decisions. Also, operational decisions can be relatively well formalized, analyzed, implemented, and reused in multiple processes executed on various platforms, cf. [86], [178], and [69]. Therefore, such decisions are best suited for integrated process and decision management. In accordance with this reasoning, in this thesis we focus only on the challenges connected with operational decisions.

According to [177], the most common examples of operational decisions in companies are *eligibility and validation decisions* (e.g., determining whether a particular person is eligible for a discount), *calculation decisions* (e.g., determining the product price), *fraud decisions* (e.g., identifying whether the insurance claim for a medical treatment is not falsified), etc. Further in this thesis we imply these types of decisions, when we provide approaches to discovery of decision models complementary to process models.

Definitions

Taking into account the operational nature of decisions relevant to business process management, we formulate the definition of decision on the basis of the DMN standard [142]:

Definition 4 (Decision). A *decision* is an act of determining an output value from a number of input values, using logic defining how the output is determined from the inputs. ◇

The concept of decision management presented in [29, 124], and [178], is aimed at promoting flexible information-driven processes and conversion of data into intelligence that enables guiding and executing enterprise decisions. The basis of decision management is the explicit representation of operational decisions in organizations. This approach implies application of decision management systems in conjunction with analytic models in order to automate, improve, and distribute decision-making capabilities across an organization [82]. Decision management also covers additional activities such as active measurement and continuous improvement of decision-making assets [178].

Definition 5 (Decision Management). *Decision management* includes concepts, methods, and techniques to support the design, administration, configuration, enactment, and analysis of operational business decisions. ◇

Decision management realizes the value of business knowledge by utilizing it to automate operational decisions [70]. Hereby, decision management systems are identified as primary mechanisms for modeling and executing decisions [34, 176]. We define the decision management system as follows:

Definition 6 (Decision Management System). A *decision management system* is a generic software system that is driven by explicit representations of operational decisions to coordinate their enactment. ⋄

According to [6, 197] and [178], the goal of decision management is to increase the speed and quality of operational decisions through their implementation, maintenance, and reuse.

2.1.3 *Integrated Business Process and Decision Management*

Operational business decisions play an important role in guiding business processes [197]. From the presented definitions of business process management and decision management, it can be observed that their ultimate goal of improving business performance is the same. Therefore, their integrated usage can deliver benefits to organizations. This has been understood by stakeholders in recent years, indicated by the appearance of a number of industrial decision service platforms in addition to existing BPM-systems, e.g., SAP Decision Service Management [6], IBM Operational Decision Management [48], Signavio [172], Camunda [2], etc. The IBM company has even formulated the principle in [48] (see p. 35) that “BPM processes must be adapted or adjusted to consider the decision management processes.” On the side of academia, a number of works dedicated to conceptual decision service platforms [30, 210] and decision ontologies [104, 145, 197] complementary to business process management techniques have been proposed. Therefore, it is evident that stakeholders need consolidated solutions on business process and decision management.

In the meantime, a paradigm of separation of process and decision concerns is also gaining strength [94, 197]. As decision making is one of the most dynamic components of businesses due to ever-changing markets, regulations and policies, agility is identified as a key criteria for successful decision automation [6, 178, 197]. It has been observed that the decision logic is often hard-coded in business process flows [197]. Such hard-coding of decisions in business process flows means that organizations often lack the necessary flexibility, maintainability, and traceability in their business operations [177].

To create a standard approach for describing and modeling repeatable decisions within organizations and to ensure that decision models are reusable in processes, the Decision Model and Notation (DMN) standard has been proposed and promoted by the OMG group [142] in recent years. The DMN standard is aimed to be complementary to the BPMN standard [138] also introduced by the OMG group for managing business processes.

In summary, we formulate the basic principle of integrated business process and decision management as follows: operational business decisions should be managed complementary to business processes of enterprises, ensuring a consistent separation of concerns.

2.2 BUSINESS PROCESS AND DECISION MANAGEMENT LIFECYCLE

As discussed above, synergy of business process and decision management systems is the emerging generation of enterprise information systems.

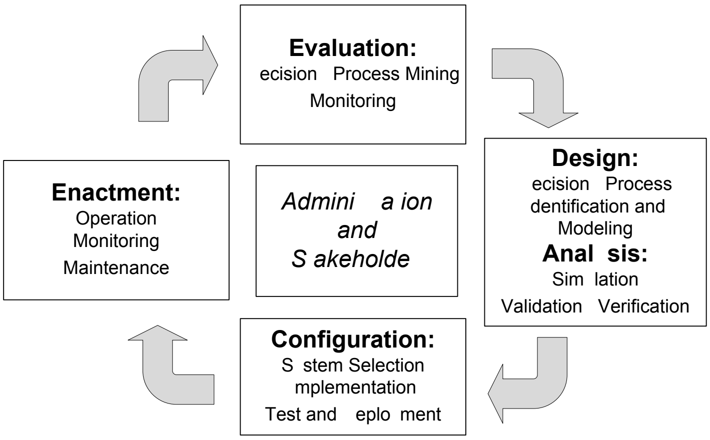


Figure 4: Business Process and Decision Management Lifecycle (BPDM-lifecycle), adapted from [202]

To provide an overall understanding of concepts and technologies behind this synergy, one needs to look into the challenges corresponding to different phases of process and decision management carried out at the enterprise. To achieve this it seems practical to view business process and decision management within the frames of a corresponding *Business Process and Decision Management lifecycle (BPDM-lifecycle)* as shown in Figure 4. The phases of the lifecycle in Figure 4 correspond to the phases of a standard business process lifecycle presented by, for example, [202]. Thereby, the phases of decision management at enterprises match up to the phases of the BPDM-lifecycle as discussed below.

The BPDM-lifecycle starts with the *Design and Analysis* phase, in which business processes and decisions are identified, validated, and modeled. This phase can be related to the phase of knowledge acquisition in decision management. Next, during the *Configuration* phase, the architecture of implementation platforms for business processes and decisions is realized in implementation platforms. Once the configuration phase is completed, business processes and corresponding decisions can be enacted, which happens in the *Enactment* phase. The *Evaluation* phase of the BPDM-lifecycle uses information available to evaluate and improve business process and decision models and their implementations. The *Administration and Stakeholders* are all kind of BPDM-users, who include developers, designers, knowledge workers, participants, engineers and owners of business processes and decisions.

The phases of the BPDM-lifecycle are organized in a cyclical structure showing their logical dependencies. It is important to note that the evolution of process and decision management at real enterprises is not necessarily simultaneous. To validate this consideration, we interviewed our partner from the European banking industry about the process of granting private loans. It turned out that their process of deciding on a loan is modeled and executed with the help of an automated credit assessing engine. However, their actual business process of receiving and handling the client application exists only ex post without being explicitly modeled or enacted with the help of any business process management system. Further interviews with other partners show that incremental and evolutionary approaches involving concurrent activities in the lifecycle phases from Figure 4 are not uncommon. In summary, the dependencies between the phases of the BPDM-lifecycle from Figure 4 do not imply a strict temporal ordering prescribing in which the phases need to be executed.

Since we are interested in discovering decision models originating from process models and event logs (cf. Chapter2), in this thesis we refer to the phases of *Design and Analysis*, *Enactment*, and *Evaluation*. The Configuration phase is beyond the scope of this thesis.

2.3 BUSINESS PROCESS AND DECISION MODELS

The BPDM-lifecycle is entered in the *Design and Analysis* phase, in which business processes and decisions are identified, reviewed, validated and represented by corresponding models. Thus, we next present the concepts that are fundamental for business process modeling, decision modeling, and separation of process and decision concerns.

2.3.1 *Business Process Modeling*

Business processes consist of a partially ordered set of related activities whose coordinated execution realizes some business goal [202]. Business process management relies on the concept of business process models, representing business processes. Here, process models are an abstraction of the real world. Therefore, process modeling helps understand processes in focusing on their core aspects, and, in turn, sharing this understanding with process participants [63]. At run-time, a process instance represents a process model. Thus, during enactment, a process model serves as a blueprint for a set of business process instances with a similar structure.

Business processes can be specified in process models with the help of graphical and formal constructs. We therefore motivate the usage of the BPMN standard for modeling processes and present its basics. Afterwards, we present the formalisms of process modeling.

Business Process Model and Notation

According to [91], the five top-ranked perceived benefits of business process modeling are (1) process improvement, (2) understanding, (3) communication, (4) model-driven process execution, and (5) process performance measurement. In order to reap these benefits, it is crucial that companies utilize process models that are captured in an appropriate modeling language [126].

To express process models, there needs to be a notation that identifies and formalizes the properties of business processes. Several modeling notations to capture business process models exist, cf. the survey in [117]. For example, flow charts have been used to model algorithms since the introduction of computers [78]. Candidates for modeling business processes are, among others, UML activity diagrams [140], event-driven process chains [170], or the Business Process Model and Notation (BPMN) [138]. Additional candidates from academia are Petri nets [167], YAWL [185], and ADEPT2 [155].

In this thesis, we use BPMN to express process models as it is de facto standard language today for business process modeling, cf. [60, 173, 197].

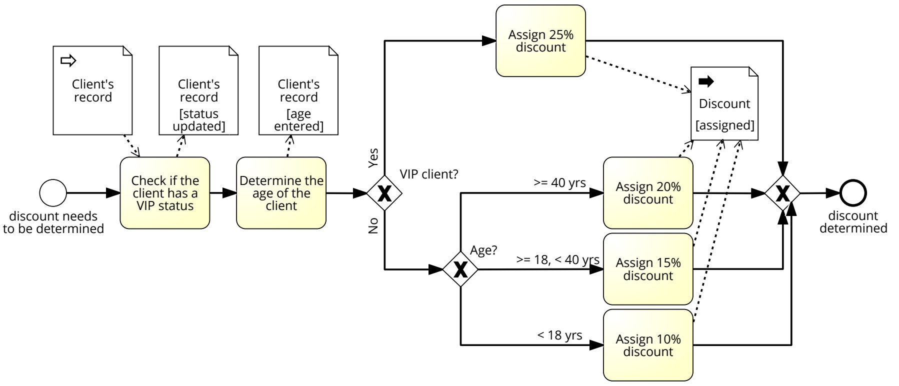


Figure 5: BPMN process model demonstrating an example of assigning a client discount: the decision logic is embedded in process control flow

Figure 5 shows a process model demonstrating an example of assigning a client discount. The notation used to express this process model is BPMN:

- The process starts and ends with the occurrence of *event models* that are represented by circles; the final *event model* is represented by a bold circle.
- Units of work conducted in a business process are represented by process *activities* depicted as rectangles with rounded edges.
- Splitting and merging of process control flow is represented by *gateways* depicted as diamonds with the appropriate markers inside, e.g., marker X stands for choice.
- The temporal relationships between activities, event models, and gateways are represented by solid directed edges.
- A process operates on *data nodes* represented by rectangles with folded upper-right corners, the behaviour of which is represented by states. data nodes provide information about what activities are required to be performed or what they produce. *Data inputs* or *data outputs* are marked with white or black arrows, and they provide the same information for processes.
- The read and write relationships between activities and data nodes are represented by dashed directed edges.

In this example, a business event happens which requires determining a client discount. Once this event occurs, a company worker checks the client's status, which is represented by the *Check if the client has a VIP status* activity and the edge connecting the start event model to this activity. The *Client's record* data node serves as an input data, and the first process activity updates this data node with state *[status updated]*. Analogously, the worker conducts a further check represented by the *Determine the age of the client* activity. Correspondingly, the *Client's record* data node is updated with the *[age entered]* state. Afterwards, the *VIP client?* gateway node is used to decide whether to proceed with activity *Assign 25% discount*, or not. If the *No* branch is activated, the *Age* gateway is used for deciding which activity should be executed next: *Assign 20% discount*, *Assign 15% discount*, or *Assign 10% discount*. When either of the activities on assigning the discount is completed, a corresponding discount value is entered into the *Discount* output data node, and its state is updated to *[assigned]*. Afterwards, the merge gateway is activated and the process completes with the final event.

Overall, BPMN is a fairly complex notation that consists of more than 100 symbols. To provide guidance to the reader, in Figure 6 we present a subset of BPMN symbols used in this thesis. The full list of BPMN elements can be found in the standard [138].

The first column in Figure 6 presents *activities* that lie at the core of the process modeling in BPMN. Activities represent atomic unit of works that are performed within a business process.

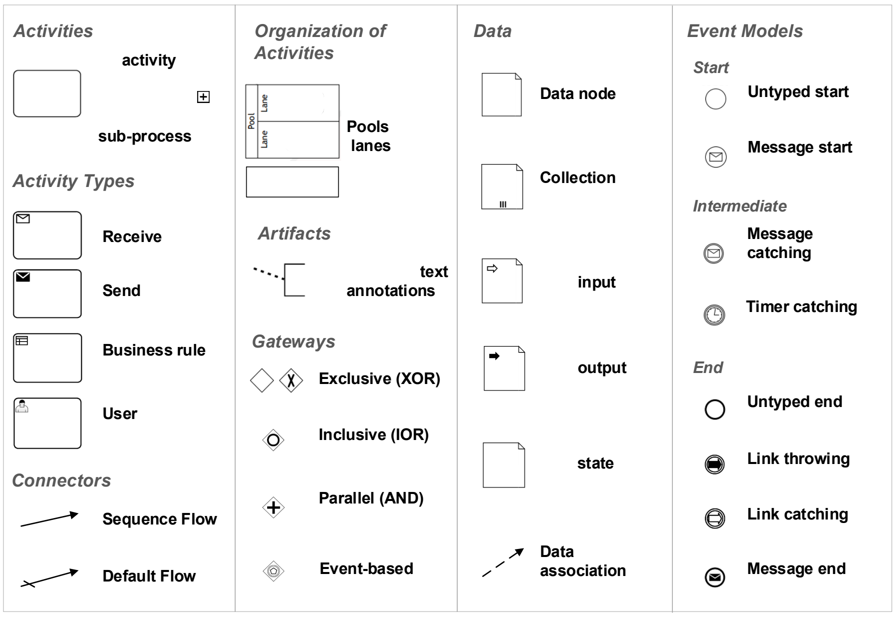


Figure 6: A subset of BPMN elements used in this thesis

The nature of the activity to be performed is specified by *activity types* as shown in the figure. A *receive* activity is an activity that is designed to wait for an information to arrive from an external process participant. Once this information has been received, the activity is completed. A *send* activity is an activity that is designed to send an information to an external participant. Once this information has been sent, the activity is completed.

Two other types of activities that are especially important for decision-intense processes are *business rule* and *user* activities. According to BPMN, a *business rule* activity provides a mechanism for the process to provide input to a business rule engine and to get the output of calculations that the business rule engine could provide [138]. A *user* activity is executed by a human performer. In this thesis, we only consider such user activities that are involved in decision making.

Activities are connected to each other using either a *sequence flow connector*, or a *default flow connector*, visualizing the rule that the branch it is representing should be chosen if all other conditions evaluate to false.

In the second column, pools, lanes, artifacts and gateways are presented. *Pools* and *lanes* are a visual mechanism to organize and categorize activities according to resources that carry out the actual work. A *pool* represents a major process participant. A pool can contain one or more *lanes* that are used to organize and categorize activities within a pool according to function or role. *Artifacts* allow developers to bring additional information into the model. In our thesis, we use BPMN artifacts such as text annotations. *Gateways* control the business process flow. A *split exclusive (XOR) gateway* routes the sequence flow to exactly one of the outgoing branches. A *merge exclusive (XOR) gateway* awaits one incoming branch to arrive in order to complete it – before triggering the outgoing flow that follows the gateway. Exclusive gateways can be shown with or without the "X" marker. A *split inclusive (IOR) gateway* routes the sequence flow to one or more of the outgoing branches. A *merge inclusive (IOR) gateway* awaits its completion of all activated branches – before triggering the outgoing flow that follows the gateway. A *split parallel (AND) gateway* branches the sequence flow by activating all outgoing branches simultaneously. A *merge parallel (AND) gateway* awaits its completion of all activated branches – before triggering the outgoing flow. An *event-based gateway* is always followed by catching event models (see the right most column in Figure 6) or receive activities, and it routes the sequence flow to the subsequent event model or activity whichever happens first.

The next column presents the *data nodes* showing which data is required or produced in an activity. This is visualized correspondingly by *data associations*. The *data collection* symbol shows that a data node clusters information collected within a business process. The *data input* symbol represents data requirements that activities in the whole business process depend on. The *data output* symbol demonstrates information produced as the result of a business process.

The last column of the figure represents process *event models*. The *untyped start or end event model* symbols signals the first or last step of a process correspondingly. Message event models carry information in processes: a *start message event model* triggers the process, an *intermediate message event model* facilitates intermediate processes, and an *end message event model* finishes the process. A *timer event model* visualizes a date or recurring time that intermediates processes by delaying it or scheduling it for a certain time. *Link event models* serve as connectors between process sections. Hereby, two corresponding link event models equal a sequence flow: one of them is a *link throwing event model*, and another one is a *link catching event model*.

Although BPMN is widely used in practice, in this thesis we use it mainly for demonstrating the developed algorithms for discovery of decision models complementary to process models based on concrete visual examples. However, since we do not want to limit the generality of our work, we next introduce the formalization of the business process model including its data perspective. Both notions are essential for deriving decision models from process information. Thereby, the work presented in this thesis can be applied to a wider range of process modeling notations.

Process Models: Formalization

In this thesis, we consider that business processes are modeled in an imperative style, which means that a process model prescribes *how* and *in which sequence* a set of activities should be executed. BPMN process model from Figure 5 is an example of a process model designed in an imperative style.

Our understanding of a process model is aligned with the definition given by Weske (see [202], p. 91) enriched with data and resource perspectives. We consider these two extra perspectives as they are a valuable source of process-related information. We utilize them later in the thesis for extracting decision models complementary to process models. Thus, a process model is defined by us as follows.

Definition 7 (Process Model). A *process model* is a tuple $pm = (N, DE, TA, \Sigma, C, F, Z, H, \alpha_s, \alpha_t, \alpha_g, \beta, \xi, \psi)$, where:

- $N = T \cup E \cup G$ is a finite non-empty set N of control flow nodes which consist of mutually disjoint sets T of activities, E of event models, and G of gateways. $E_B \subseteq E$ is the set of boundary events. Function $\alpha_s : T \rightarrow \{task, subprocess\}$ distinguishes activities into atomic tasks and non-atomic collapsed subprocesses. Thereby, function $\alpha_t : T \rightarrow \{abstract, manual, user, business\ rule, service, script\}$ assigns to each activity a specific type. Function $\alpha_g : G \rightarrow \{XOR, IOR, AND, event\text{-}based\}$ assigns to each gateway a type in terms of a control flow construct. Function $\beta : T \nrightarrow 2^{E_B}$ assigns sets of boundary events to activities.
- $C \subseteq N \times N$ is a finite set of directed flow edges, which define the control flow. Σ is a finite set of conditions. $G_o \subseteq G$ is the set of gateways with multiple outgoing edges, such that $(\alpha_g(g) = XOR \vee \alpha_g(g) = IOR)$, $g \in G_o$. Function $\xi : (G_o \times N) \rightarrow \Sigma$ assigns conditions to control flow edges originating from gateways with multiple outgoing edges, where for $\forall g \in G_o$: $\exists\ n_1, n_2 \in N, n_1 \neq n_2$ such that $(g, n_1) \in C \wedge (g, n_2) \in C$.
- $DE = DN \cup DS$ is a finite set of data elements, consisting of mutually disjoint sets DN of data nodes, and DS of data stores. $F \subseteq (DE \times T) \cup (T \times DE)$ is the set of directed flow edges, indicating read respectively write operations of an activity with respect to a data node, which define the data flow.
- TA is a finite set of text annotations. $Z \subseteq (TA \times T) \cup (T \times TA)$ is the set of symmetric associations that connect activities with text annotations.
- H is a finite set of resources. Herewith, function $\psi : T \longrightarrow H$ assigns a corresponding resource to each activity. ◇

An example process model is given Figure 5, and its elements are discussed in the paragraph below the figure.

In some cases, it is reasonable to consider a subset of the process model elements. It might be useful for analyzing a part of a process model with the focus on a certain aspect of it, e.g., a decision point containing several control flow splits and the activities involved in this decision. In this thesis, we refer to such groups of elements as *process fragments*, which are defined by us in Definition 8.

Definition 8 (Process fragment). Let $pm = (N, DE, TA, \Sigma, C, F, Z, H, \alpha_s, \alpha_t, \alpha_g, \beta, \xi, \psi)$ be a process model. A *process fragment* $pf = (N', DE', TA', \Sigma', C', F', Z', H', \eta_s, \eta_t, \eta_g, \kappa, \sigma, \zeta)$ is a connected subgraph of process model pm such that $N' \subseteq N$, $DE' \subseteq DE$, $TA' \subseteq TA$, $\Sigma' \subseteq \Sigma$, $C' \subseteq C$, $F' \subseteq F$, $Z' \subseteq Z$, and $H' \subseteq H$. Functions η_s, η_t, η_g, κ, σ, and ζ are restrictions of functions α_s, α_t, α_g, β, ξ, and ψ respectively with corresponding new domains. ◇

For example, Figure 7 is a process fragment of the process model from Figure 5. This process fragment focuses on the decision point reflecting the steps of the decision on assigning a certain type of discount.

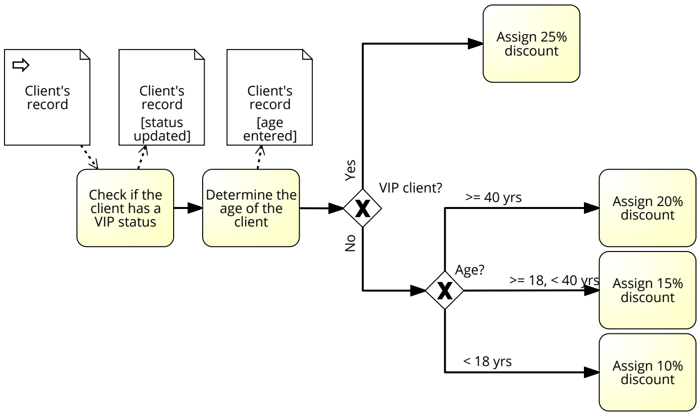


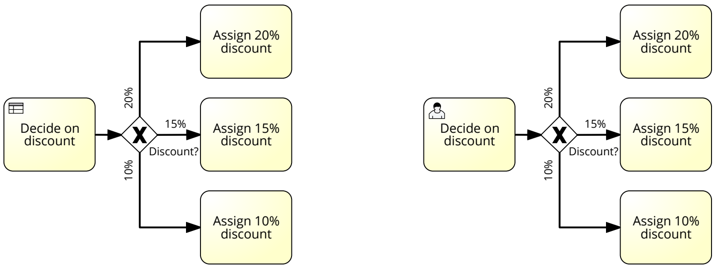
Figure 7: Example process fragment of the process model from Figure 5

Another subset of the process model elements, that is relevant for the work presented in this thesis, are process activities that carry decisional value. Adhering to BPMN, we consider that the business rule and user activities in a process model serve as a natural placeholder for process decisions. The DMN standard also recommends that decision activities are executed either manually as by a user, or in a business-rule-driven manner. We consider both types of decision activity, as reflected in Definition 9.

Definition 9 (Decision Activity). Let $DA \subseteq T$ be the set of decision activities for a given process model pm. If $da \in DA$ is a *decision activity*, then $\alpha_t(da) \in \{business\ rule, user\}$. ◇

For example, Figure 8 shows two process fragments that represent variants of incorporating of a decision activity *Decide on discount*. Figure 8a shows the decision activity of a business rule type, where the discount is assigned, for example, by a business rule engine. Figure 8b shows the decision activity of a user rule type, where the decision on discount is taken by a human performer.

The activities of a process model operate on an integrated set DN of data nodes, which represent application data created, modi-



(a) Decision activity of a business rule type

(b) Decision activity of a user type

Figure 8: Example process fragments showing two variants of a decision activity *Decide on discount*

fied, and deleted during the execution of a process model. The term data flow refers to data dependencies between process activities and data. An example of a process model is given in Figure 5, followed by description of its elements.

In our work, we use the distinction of process data into data classes and data nodes (see Figure 9), which can be viewed as analogue to the object-oriented programming paradigm. *Data class*, used in a business process, serves as an abstract data type, which describes the properties of *data nodes*. The data nodes can be viewed as instances of the data classes at the modeling level. Each data node is associated with exactly one data class in a process model. The particular values of data class properties are assigned to the data node associated with it.

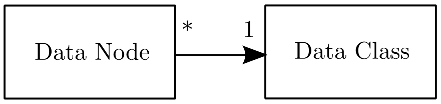


Figure 9: Relations between data entities

Definition 10 (Data Class). A *data class* is a tuple $dc = (name, S, J_c)$, where:

- S is a finite set of data states that dc has;
- J_c is a finite set of attributes that dc has, where each attribute $j \in J_c$ has a corresponding domain $Dom(j)$ represented by a set of either numeric or nominal values. ◇

Definition 11 (Data Node). Let DC and DN be the sets of data classes and data nodes associated with a given process model pm correspondingly. Each *data node* $dn \in DN$ is a tuple $dn = (name, s, \delta, \tau, J, T_{dn}, F_{dn}, H_{dn}, \varphi)$ related to a data class $dc \in DC$, where:

- *name* is a constant labeling the data node dn, which also serves as a reference to dc;
- $s \in S$ is a variable indicating the state assigned to dn, where S is the set of data states of dc;
- $\delta : DN \longrightarrow \{singlinst, multinst\}$ is a function indicating if $dn \in DN$ is a collection (multinst) or not (singlinst);

- $\tau : DN \longrightarrow \{input; output; default\}$ is a function indicating if $dn \in DN$ is an input data node (existed before the start of the process), an output data node (will exist after termination of the process) or none of these (default);
- $J \subseteq J_c$ is a set of attributes assigned to dn, where J_c is the set of attributes of dc;
- $T_{dn} \subseteq T$ is a set of process model activities which are related with the data node dn by the respective data flow $F_{dn} \subseteq F$, such that $F_{dn} = (dn \times T_{dn}) \cup (T_{dn} \times dn)$. Function $\varphi : DN \longrightarrow 2^{H_{DN}}, H_{DN} \subseteq H$, assigns to $dn \in DN$ a power set of resources responsible for execution of the activities that are in the data flow relation with dn. ◇

We assume that the resource executing an activity is either allocated to the data node read or write by this activity. Also, as it can be seen from Definition 11, the set of attributes J store the context data relevant to the business process, i.e., the particular characteristics of the data class. An example data class for the process model from Figure 5 is the *Client's record*. The example data nodes for the same process model are the concrete occurrences of the data class *Client's record [status updated]* and *Client's record [age entered]*.

2.3.2 *Decision Modeling*

As previously stated, a decision is an act of choosing one or a set of possible outcomes based on a set of inputs [142]. We focus on operational decisions that are repeatable, have a business value, and occur in technical and informational environment of companies. Decision modeling is needed to systematically describe decisions, so that they are understandable to decision participants, and to continuously improve them [178]. Below, we motivate the usage of the DMN standard and present basics of this notation, which is followed by a presentation of the decision modeling formalisms.

Decision Model and Notation

In [197], Halle et al. state that business logic of activities involved in decision making is often not documented in BPMN process models. At the same time, operational decisions should be carefully modeled since they are a subject of deliberate attention during compliance checks. Providing a common visual language would allow an understanding of decision making and a sharing of this understanding between decision participants. In addition, visual decision notation would serve as a tool for identification of issues, thereby providing an opportunity for decision improvement.

To create a standardized bridge for the gap between business decision design and its implementation, the OMG group released in

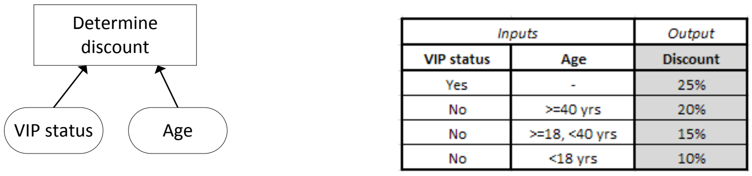
2015 the DMN standard. One of the main goals of DMN is to provide a common language which is understandable to business users, such as the business analysts who need to create models of the decision making process, the technical developers who are responsible for automating process decisions, and, finally, the business people who will manage and monitor these decisions [142]. DMN provides constructs that are needed for modeling decision-making in processes, that can be automated optionally. The DMN standard is created in such a way that it can be used complementary to other established industry standards such as BPMN, Unified Modeling Language (UML), Entity–Relationship model (ERD), Case Management Model and Notation (CMMN), Business Motivation Model (BMM), etc.[178].

The DMN standard was developed by the OMG Group through the contribution of such companies as SAP, IBM, Microsoft, HP, Oracle, and other major companies [136]. Although DMN is a young standard, it is gaining acceptance in industrial, and academic worlds. The Decision Management Community [46] provides a catalogue of vendors providing DMN support tools, which currently lists 16 vendors across a range of countries and continents. These vendors include AlfrescoActiviti, Blueriq, Camunda, DecisionsFirstModeler, OpenRules, FICO, Drools, Sapiens, Signavio, Trisotech, etc. The DecisionsFirstModeler CEO J. Taylor recommends DMN as "the best approach to decision modeling available today," because it is an open, not-for-profit standard for unambiguous, concise, and complete decision management at enterprises [178]. Also, a range of scientific works is dedicated to exploration of the standard properties, e.g. research on its visual expressiveness [50], possibilities of enhancing declarative process models with DMN logic [127], integration of DMN and other OMG standards [101], etc.

DMN defines two levels for modeling decision logic, the *decision requirements level* and the *decision logic level*. The first one represents how *decisions* depend on each other and what *input data* is available for the decisions. Therefore, these nodes are connected with each other through *information requirement edges*. A decision may additionally reference the decision logic level where its output is determined through an undirected association. The decision logic level describes the actual decision logic applied to take the decision. Decision logic can be represented in many ways, e. g.by an analytic model, a textual description, or a decision table.

Figure 10 shows an example decision model. The DMN Decision Requirements Diagram is the visual notation used to express this model (see Figure 10a):

- Rectangles represent decisions referred to as *decisions nodes* required to be taken. Each decision generates a set of results which are using a set of inputs, i.e., other decision nodes and data nodes (see the next bullet point).



Inputs		Output
VIP status	Age	Discount
Yes	-	25%
No	>=40 yrs	20%
No	>=18, <40 yrs	15%
No	<18 yrs	10%

(a) DMN decision requirements diagram (DRD)

(b) Decision table

Figure 10: DMN decision model corresponding to the example of assigning a client discount

- A *data node* is represented by an ellipsis and is an area of data required to take the decision. According to [70], this might be: (1) specific data relating to the case being processed (such as application, account, or claim data); (2) contextual data relating to the conditions surrounding the case (such as the prevailing base interest rate).
- Solid arrows represent dependency requirements referred to as *information requirement edges*, such that a piece of information (a decision or a data node) is used in a decision. The information requirement edges are not conditional; the arrow must be present if the piece of information is always required to take the decision.

The decision requirements diagram does not visualize the decision logic used for determining DRD output, but a corresponding decision table (see Figure 10b) is referenced through an undirected association corresponding to the decision node. The example shows the same decision logic as it is used in the process model in Figure 5. In particular, the decision to be taken refers to the *discount* given to a customer. The corresponding logic is defined in the associated decision table *manage discount table*. Information about *VIP status* and *Age* needs to be considered and the result of this decision is referenced in the corresponding decision table.

To provide guidance to the reader, in Figure 11 we present a subset of DMN symbols used in this thesis. The full list of DMN elements can be found in the standard [142].

As reflected in Figure 11, the decision requirements level is described in DMN by a Decision Requirements Diagram (DRD), which consists of DRD elements and their interdependencies. The following DRD elements are presented in the first column in Figure 11:

- A *decision* is an act of determining an output from a number of inputs, using decision logic (cf. Definition 4). Therefore, decisions are acts on process-related data.
- A *business knowledge* denotes any decision logic which is capable of being represented as a function encapsulating business know-how, e.g., in the form of a decision table. Hence, the busi-

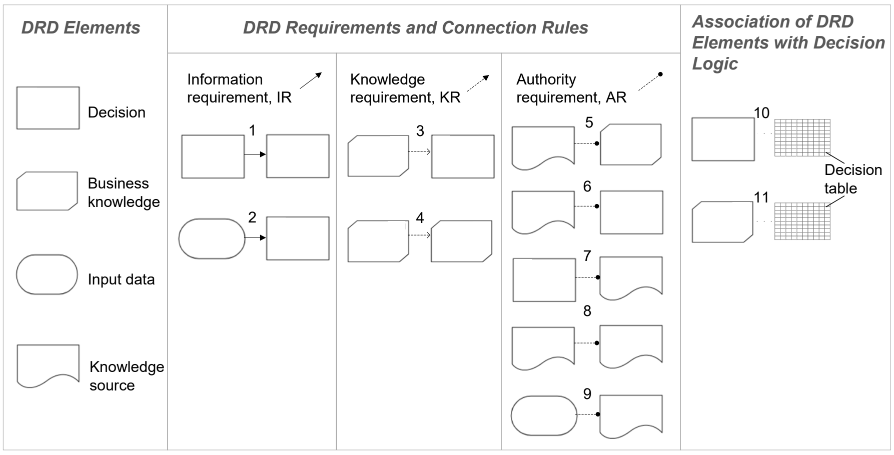

DRD Elements
DRD Requirements and Connection Rules
Association of DRD Elements with Decision Logic
Decision
Business knowledge
Input data
Knowledge source
Information requirement, IR
Knowledge requirement, KR
Authority requirement, AR
1
2
3
4
5
6
7
8
9
10
11
Decision table

Figure 11: A subset of DMN elements used in this thesis

ness knowledge elements represent functional requirements for acting on process-related data.

- An *input data* denotes the information used as an input by one or more *decisions*. Thereby, the input data elements directly represent process-related data.
- A *knowledge source* denotes an authority for a business knowledge or a decision, which can be represented as the domain experts responsible for maintaining decisions or the source documents from which decisions are derived. Thus, the knowledge source elements represent non-functional requirements for acting on process-related data.

As can be seen from the central three columns in Figure 11, the dependencies between DRD elements express three kinds of requirements:

- An *information requirement* denotes either a decision output being used as input to another decision (rule 1), or an input data being used as an input to a decision (rule 2). There is no distinction in DMN between input data that is always required and input data that is sometimes required. Thus, information requirements may be optional when the decision is made for a specific transaction. Thereby, information requirements express functional requirements for acting on process-related data.
- A *knowledge requirement* denotes the invocation of a business knowledge node by the decision logic either of a decision (rule 3), or of another business knowledge (rule 4). In this way, knowledge requirements also represent functional requirements for acting on process-related data.
- An *authority requirement* denotes the dependence of a DRD element on another DRD element that acts as a source of guidance or knowledge (rules 5, 6, 7, 8, and 9). This might be used to record the fact that a set of business rules must be consistent with a published document (e.g., a piece of legislation or a statement of business policy), or that a specific person or orga-

nizational group is responsible for defining some decision logic. As a special case, they may be drawn from the input data and decision nodes to the knowledge source nodes (rules 7 and 9). Such a modeling approach is recommended by DMN in the situations if a source of business rules was originally built through data mining or some other kind of data analysis. Thus, authority requirements represent non-functional requirements for acting on process-related data.

The second logic layer of DMN is decision logic which specifies *how* decisions are to be taken. Decision logic is not visually presented in the DRD diagram, but it is added to a DMN decision model through a link with either the decision, or the business knowledge nodes of a DRD diagram. As visualized by the last column in Figure 11, such links represent undirected association of corresponding DRD elements with decision logic. If a decision requires an input data, the value of the variable is assigned the value of the data source attached to the input data at execution time. One of the most widely used representation for decision logic is a decision table [142, 191, 194].

In this thesis, we present all visual examples of decision models with the help of the DMN standard. However, to not limit the generality of our work on the discovery of decision models complementary to process models with the DMN standard, we introduce next the decision model formalization. Thus, the work presented in this thesis can be applied on a wider range of decision modeling notations.

Decision Models: Formalization

Our formalism of a decision model is aligned with the DMN standard [142]. According to DMN, a decision model consist of two logical layers: (1) decision requirements; and (2) decision logic, e.g., represented with the help of decision tables. We consider these two layers below.

The first logical layer of the DMN decision model is the decision requirements layer. People are much better at extracting information from figures than from text, which is why decision requirements diagrams are recommended [70]. Decision requirements diagrams can be viewed as a decomposition of process decision-making into a set of interrelated decisions and areas of supporting information. We formally define the decision requirements diagram as follows.

Definition 12 (Decision Requirements Diagram). A *decision requirements diagram* (DRD) is a tuple $drd = (D, BK, ID, KS, IR, KR, AR)$ consisting of:

- a finite non-empty set of *decision* nodes D;
- a finite set of *business knowledge* nodes BK;
- a finite non-empty set of *input data* nodes ID;
- a finite set of *knowledge source* nodes KS;

- a finite non-empty set of directed edges IR representing *information requirements* such that $IR \subseteq (ID \cup D) \times D$;
- a finite set of directed edges KR representing *knowledge requirements* such that $KR \subseteq BK \times (D \cup BK)$;
- a finite set of directed edges AR representing *authority requirements* such that $AR \subseteq (D \cup ID \cup KS) \times (D \cup BK \cup KS)$.

Herewith, $(D \cup BK \cup ID \cup KS, IR \cup KR \cup AR)$ is a directed acyclic graph (DAG). ◇

An example decision requirements diagram is presented in Figure 10a, and its elements are discussed in the paragraph above the figure.

The decision nodes of DRD provide a clear but succinct definition of the functional requirements for decision making. The input data nodes identify all the types of data required for decision taking to be made available to decision nodes and modeled within them. As Fish points out in [70], the DRDs are useful, because they allow the broad scope and structure of the decision-making to be appreciated at a glance and discussed by all the participants of the decision-making process, such as the project managers, business domain experts, analysts, designers, developers, and testers.

The most recent version of the DMN standard introduced a new layer of a decision model, the *decision service* layer, that is a visual mechanism to reflect which nodes of DRD should be implemented as services. The decision service layer is beyond the scope of the current thesis, as we primarily focus on creating an explanatory decision model that can be discovered from the existing knowledge about a process (cf. Section 1.2). Thereby, the implementation of the discovered model is out of the scope of our work.

Besides DMN, there also exist other methods and notations to specify requirements for decision making. UML use case diagrams [140] treat the system as a black box rather than exposing the structure of its decision-making. Entity relationship diagrams can be used for capturing knowledge structure [197], but they are more useful for modeling specific rules than high-level requirements. Therefore we adhere to the DMN decision models.

A decision may additionally reference the decision logic level, the second layer of the DMN decision model, where its output is determined through an undirected association. One of the most widely used representations for decision logic is a decision table, which we utilize in the rest of this thesis.

The decision table representation is one of the most widely used formats for representing decision logic. Decision tables incorporate all necessary components for decisions, such as inputs, outputs and decision rules. What is crucially important, is that they are easily understood by business users [70, 142]. According to DMN, further representations of decision logic can be used, e.g., with the help of the text documents or computer programs. However, we leave such representations of decision logic out of the scope of the current thesis.

Our understanding of a decision table conforms to DMN [142]. According to the standard, a decision table describes the relation between a set of input values and a set of output values. Each decision table input has a domain over which logical expressions can specify conditions. During run-time execution, each input is assigned a value, and if the conjunction of logical expressions of all inputs evaluates to true, the corresponding output values are yielded. The output values belong to the domain of the decision table output. The possibilities of relating different inputs to outputs are represented in a tabular manner such that each row of the table corresponds to a rule. Thereby, in this thesis we consider the most usual case where decision table has only one output variable. If rules overlap and multiple rules can match, a hit policy indicates how to determine the output.

In Definition 13, we provide the formal definition of a decision table. The definition is followed by a decision table example and a discussion of the hit policies that can be encountered in this thesis.

Definition 13 (Decision Table). A *decision table dt* of the set of decision tables DT is a tuple $dt = (I, O, R, \chi)$, $dt \in DT$, where:

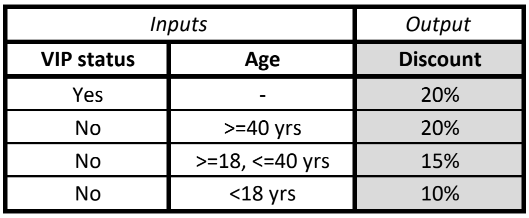
- $I = \{I_1, ..., I_v\}$, $v \in \mathbb{N}^+$ is a finite non-empty set of input variables *(inputs)*;
- O is an output variable *(output)*;
- $R = \{R_1, ..., R_n\}$, $n \in \mathbb{N}^+$ is a finite non-empty set of mappings (*decision rules*), which relate a subset of inputs to an output:

$$\forall i \in [1;n]\ R_i : \bigwedge_{j=1}^{w} (I_j\ op_j\ q_j) \longrightarrow Dom(O), 1 \leq w \leq v \qquad (1)$$

  where $op_1, ..., op_w$ are comparison predicates, $q_1 \in Dom(I_1)$, ..., $q_w \in Dom(I_w)$ are constants representing values from the domains of the inputs, and $j, w \in \mathbb{N}^+$;
- $\chi : DT \longrightarrow \{unique, collect\}$ is a function that assigns each decision table $dt \in DT$ a hit policy.

◇

An example decision table is presented in Figure 12. The inputs in this decision table are *VIP status* and *Age* which are determined for a client during decision execution. The output of this decision table is *Discount* assigned to the client. There are four decision rules constituting this decision table which specify the decision logic about how the discount is assigned to the client. A decision rule example for this decision table is the last rule of the table:



Inputs		*Output*
VIP status	**Age**	**Discount**
Yes	-	20%
No	>=40 yrs	20%
No	>=18, <=40 yrs	15%
No	<18 yrs	10%

Figure 12: Decision table which contains: (1) overlapping rules; (2) rules that have the same output.

$$VIP\ status{=}No,\ Age{<}18\ yrs \longrightarrow Discount{=}10\ \% \tag{2}$$

For a given set of input values, the matching rules of a decision table indicate the resulting value for the output name. Thereby, decision tables are allowed to contain overlapping decision rules, where rules match for the same set of input values but do not necessarily map to the same output values. For example, the second and the third rules in the table from Figure 12 are overlapping.

For the case when rules overlap and multiple rules can match the run-time input, DMN defines different types of hit policies that indicate how to determine the output. Thus, the *single-hit* policies return the output of one rule only, and the *multi-hit* policies return a list (or sequence) of outputs. Hit policies can be assigned to decision tables by a process expert, with respect to a relevant business context. As indicated in Definition 13, in this thesis we use only *unique* and *collect* hit policies, which are described below. The full list of the DMN hit policies can be found in the standard [142].

The first type of hit policies which we use in this thesis are the *unique* policies that are of the single-hit type. If the unique hit policy is assigned to a decision table, no overlap of rules is possible and all rules are disjoint. This is the default hit policy which is used for the most of decision tables in this thesis, unless specified otherwise in the text. For example, the table from Figure 10b serves as an example of a decision table with non-overlapping rules that has the unique hit policy. Then, for an example input $VIP\ status{=}No,\ Age{=}30\ yrs$, the output $Discount{=}15\%$ is returned.

The second type of hit policies which we use in this thesis are the *collect* policies that are of the multi-hit type. If the collect hit policy is assigned to a decision table, the decision table rules can overlap, and the output result is a list of all the matching output values. For example, the table from Figure 12 contains overlapping rules, and the collect hit policy can be assigned to it. Then, for an example input $VIP\ status{=}No,\ Age{=}40\ yrs$, the output $Discount{=}\{20\%, 15\%\}$ is returned.

We use a collective term *decision model* for an aggregate of a decision requirements diagram *drd* and a corresponding decision table *dt*. An example decision model is presented in Figure 10, and its elements are discussed in the paragraph below the figure.

A decision table is considered *complete* if for each possible combination of input values at least one rule is matched. If a decision table is *incomplete*, and is called with an input combination that it does not consider, the result is undefined, which may lead to undesired behavior. The decision table from Figure 10b is an example of a complete decision table.

Adhering to the view of the OMG group on the nature of process and decision modeling, we consider that business processes are modeled in an imperative style, whereas business decisions are modeled in a declarative style. As discussed in Section 2.3.1, an imperative style of business process modeling means that a process model prescribes *how* activities should be executed. In contrast, a decision model instead expresses the *decision requirements and logic* without describing control flow of decision making. This view is supported, for example, by J. Taylor et al. in [177, 178] and by A. Fish in [70], which is that decision modeling should focus on *what* is to be done, not *how* it should be done.

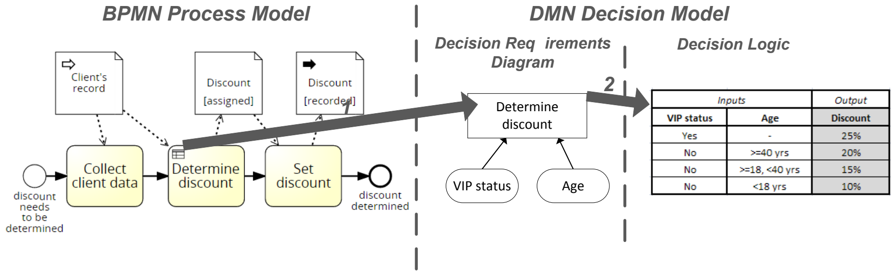
2.3.3 *Separation of Process and Decision Concerns in Modeling*

Below, we present two examples demonstrating how separation of process and decision concerns can be accomplished. The first example refers to the business process represented by process model from Figure 5 and the decision model from Figure 10. The second example is a real-world use case demonstrating more complicated process and decision models, and their connection. Next, we introduce principle of separation of concerns and its properties. Afterwards, we provide a definition of a decision-aware process model.

Introductory Example: Discount Calculation

Consider again the business process model in Figure 5. While the described process is reasonably easy to understand, it is clear that control flow that incorporates decision logic can become complex upon expansion of underlying business rules. If a new business rule is introduced, e.g., to take the client's nationality and loyalty into account, then a new nest of gateways and branches is required. Such a modeling approach hinders agility of processes, since changing decision logic requires changing the process control flow which is cost-intensive, especially if the process is implemented in an information system [197].

The separation of concerns principle states that decision logic should be modeled complementary to process models. In this case, this means to identify that the complete process from Figure 5 can be represented by steps of determining model and setting the discount for a customer. In Figure 13, we show the refactored process model. It references the decision model (see Figure 10) through an undirected association depicted for visualization purpose with the bold arrow 1 which is not a part of the BPMN or DMN notations. The decision model explains in details how the discount is determined. Hereby, the DRD also references the decision table through an undirected association which is depicted with the bold arrow 2.



Inputs		Output
VIP status	Age	Discount
Yes	-	25%
No	>=40 yrs	20%
No	>=18, <40 yrs	15%
No	<18 yrs	10%

Figure 13: Demonstration of paradigm of separation of process and decision concerns on example of BPMN and DMN models using the examples from Figure 5 and 10

The process model depicted in Figure 13 is an example of how a process model can be "aware" of decision making that is part of a process but that is described in a dedicated decision model. It can be seen that the process model is simplified: decision logic which was encoded in control flow structures is now "exported" into a corresponding decision model. Such representation increases process agility, because now changing decision logic can be done simply by modifying or adding new lines of the decision table without changing the process.

Real-World Example

Next, we present a real-world example communicated to us by an industry partner, a debt collection company operating throughout the European Union. Using a standardized process, the company supports customers in enforcing their claims against debtors. Within this process, numerous decisions have to be made: what kind of legal documents are needed to evaluate the claim, how to communicate with the debtors, in which jurisdiction the claim should be enforced, etc. Usually, cases take several weeks before completion. However, it a case may go to court and then it may take several years before a resolution is reached.

An excerpt from the debt collection process is represented by the BPMN process model shown in Figure 14. Some of the tasks of the process reference the corresponding DMN decision requirements diagram shown in Figure 15.

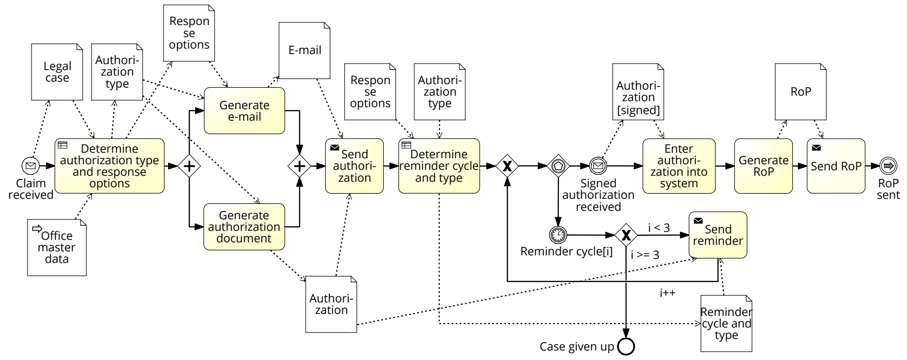


Figure 14: Excerpt from the debt collection process represented by a BPMN process model

A new process instance starts when a customer enters a claim containing general information through a company website. First, the activity *Determine authorization type and response options* is executed. This activity is typed as a business rule activity and references the corresponding decision requirements diagram from Figure 15 through an undirected association marked by a small icon of a table in the left upper corner of the activity. At this step, only intermediate decisions *Authorization type* and *Response options* of the decision requirements diagram are invoked, resulting in production of the corresponding data nodes by the activity. This correspondence is indicated by the label equalities of decisions from Figure 15 and data nodes written by business rule activities of the process model. The *Authorization type* decision determines if the company requires a Power of Attorney (PoA) or Notice of Assignment (NoA) from the customer. This information is used in the following two concurrent activities *Generate e-mail* and *Generate authorization document* of the process to customize the e-mail that is sent to the customer. The *Response options* decision determines different possibilities of how the client can provide the PoA/NoA to the company, which depends on the country of residence of the customer. For example, a personal visit is possible if the customer resides in the country where the headquarters of the company is situated, but in other cases, only mail or fax communication is possible. Next, the activity *Send authorization* is executed, sending the generated e-mail with the *Authorization* document attached to the client.

The process proceeds with execution of activity *Determine reminder cycle and type*. This is also a business rule activity. Again, the decision model from Figure 15 is invoked, and the result output is stored in the corresponding data node written by the invoking activity. This decision represents a *top-level decision* of the diagram in Figure 15, as no further decisions depend on it. Note, that this decision depends on subdecisions *Response options* and *Authorization type* that have already been evaluated before. The output results of these

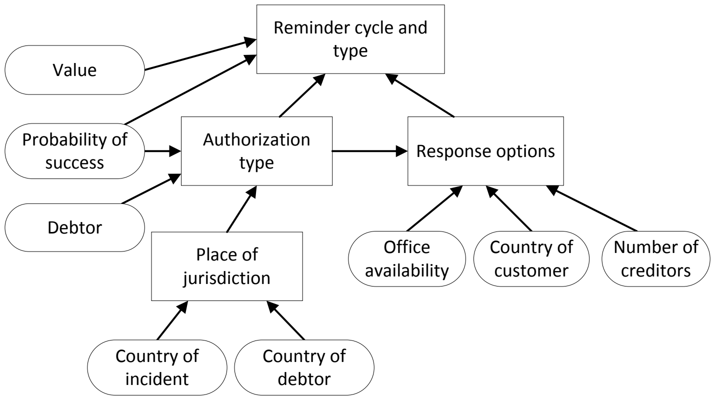


Figure 15: DMN decision requirements diagram showing decisions of the debt collection process

subdecisions are read by the business rule activity and passed along to the decision model as inputs in such a way that there is no need to evaluate them again. The corresponding activity specifies the cycle and type of reminders which are sent to customers in case do not respond to requests. For example, an activity could specify that a customer should be reminded every five days, in total three times, the first two times via e-mail, and the third time via phone. This is depicted in the figure by the condition *Reminder cycle* $[i]$ annotated to the timer event model, and the XOR-split conditions $i < 3$ and $i >= 3$ annotated to the outgoing edges of the split gateway, where i is a number of reminders sent. At this point, it can already be noted that this process model is not of a very good quality, as it incorporates the decision logic within the process model.

After sending the authorization document, the company waits for the customer's response. If it does not arrive within the time interval specified by the *Reminder cycle and type* decision, a corresponding reminder is sent out. If no answer is received after the third reminder, the case is abandoned. If the signed authorization document is received, it is stored in the company's IT system during execution of *Enter authorization into system* activity, and a request of payment (RoP) is generated and sent to the debtor during the execution of the last two activities of the process. The remainder of the real process is not shown here as the illustrated excerpt is sufficient for discussing the challenges described in this thesis. In reality, the process can go on for a long time if the debtor refuses to pay. In this case, a lawyer may eventually get instructed to settle the case in court.

Separation of Concerns: Motivation and Principle

In [178], Taylor et al. state that using a decision model complementary to a process model is one of the most powerful inter-model synergies available to business process management. Based on a literature overview of this area, we distinguish the following three motivations for modeling decisions complementary to process models:

- *Simplification of process model / reduction of complexity.* Because business decisions are identified and managed as discrete elements of the new process, the process becomes simpler: it has a single decision-making activity instead of complex decision control-flow structures in the process design [69, 142, 197]. More specifically, clusters of XOR-structures describing decision making are excluded from the process model and are imported into a dedicated decision model resulting in less complex process models [76]. Hereby, these structures can be represented as an ordered list of rules of decision tables.
- *Agility / Better change management.* Decisions are the subject of frequent change in processes [6, 29]. Decision changes occur for different reasons, such as internal change of business policies improvements (e.g., to reduce risk or improve revenues as a response to market changes), or compliance (e.g., imposed by law changes). The operational decisions must recognize the organizations' need for agility: a readiness and ability to respond quickly to changing business conditions [69]. In business, being agile means that an organization has the ability to not only sense environmental change, but also has the capabilities and processes in place to efficiently and effectively respond to that change. In [178], it is noted that making changes in decisions that are complementary to processes reduces time and costs (see example of costs reduction in Section 1.1). That said, separation of concerns allows changes in decision models without changing corresponding process models [197]. According to [70], decision tables containing business rules are particularly useful to support the agility of processes, as they can incorporate policies that change frequently and are easily updated. The ability of decision tables to support dynamic changes in environment allows modifying business process implementation without changing and redeploying it.
- *Reusage of decisions.* Separate models allow decisions and processes to be reused independently of one another [142, 197]. In [70], the authors recommend designing decision models and services complementary to process models in such a way that it is possible to use them across functions, channels, and lines of business. FICO, one of the DMN vendors, states in [69], that the same decision rules, for example, how you deal with a customer's order in a particular situation should be applied identically across your BPMS. This is true especially of compliance, where companies must be able to show consistency of behaviour. Additionally, inconsistent treatment across systems

> could even leave you accused of bias [69].

In order to clarify how a decision model should be modeled complementary to process model, a list of corresponding properties should be specified. Thereby, both process and decision models that are "aware" of each other should adhere to the separation of concerns principle. On the basis of [178], we define the decision-aware process model and process-aware decision models as follows:

Definition 14 (Decision-Aware Process Model). A *decision-aware process model* is a process model that has the following properties:

- Process activities that come to conclusions based on business logic are assigned with the *business rule* or *user type* (such activities interchangeably referred to as *decision activities*).
- Decision activities in the process model reference a corresponding decision model through undirected association, normally realized during the implementation phase.
- The process model describes *how* the data is provided to the decision model by specifying the involved control flow.
- A decision activity can invoke a decision node from the decision model at any level. ◇

Definition 15 (Process-Aware Decision Model). A *process-aware decision model* is a decision model that has the following properties:

- Each decision node in a decision model can be associated with one or more process models or activities.
- Decision nodes in the decision model are stateless and decoupled from the control flow.
- Not all decision nodes need to be explicitly invoked by a decision activity from the process model. ◇

An example of a business rule activity (or a decision activity) is the activity *Collect client data* in Figure 13. An idea of undirected association between process and decision models is illustrated in Figure 13 with the help of bold arrows 1 and 2 (the arrows are not part of the modeling notation).

The concrete examples of invocations of decision models by process models follow. For instance, the decision activity *Determine authorization type and response options* in Figure 14 invokes subdecisions *Authorization type* and *Response options* in Figure 15, and the decision activity *Determine reminder cycle and type* invokes the top-level decision in Figure 15. Decision *Place of jurisdiction* in Figure 15 is not invoked by any decision activity in Figure 14, but it is invoked by decision *Authorization type* in Figure 15, since the second decision needs the first decision as input.

Although separation of process and decision concerns (e.g., as presented in Figure 13) is prescribed by the DMN standard, most of today's business processes are not designed to be decision-aware

[18, 197]. According to [178], decision modeling is an emerging technique that has yet to be widely adopted. In many companies the state-of-the-art managing of process decisions is implemented "in an obscure fashion in some application code by technical people" (in [29], p.4). This served as our motivation for providing stakeholders with methodologies for discovery of decision models complementary to process models.

2.4 EXECUTION OF PROCESSES AND DECISIONS

Once a business process is designed and configured, it can be executed. In this section, the formal foundations are laid to cover business process and decision execution.

2.4.1 *Process Enactment*

The execution can be performed completely automatically, completely manually, or a combination of both. In this thesis, we do not differentiate between automatic or manual executions, yet, we impose certain restrictions on how the execution of a process has to be logged in order to apply the introduced techniques.

Definition 16 (Process Execution, Process Instance, Activity Instance). Given a process model pm, a process execution is a sequence of activity instances $ai_1 \dots ai_n$, $n \in \mathbb{N}^+$, where each ai_i is an instance of an activity in the set of activities T of pm. ◇

Event logs are central artifacts of process executions [184]. An event is a real-world happening occurring in a particular point in time at a certain place in a certain context [85, 119]. In [67], an event is defined as either a real world occurrence (e.g., an incoming phone call), or an observation of a real world occurrence within a particular system or domain (e.g., a missed call log at the desk phone).

In the context of business process management, the systems that store and manage information related to processes are called process-aware information systems [62, 182]. More specifically, a *Process-Aware Information System (PAIS)* is a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models (cf. [182]). As can be seen from Figure 1, PAIS are recording real-world events in event logs. PAIS include all kind of workflow management systems: ERPs, CRMs, rule-based systems, etc.

In our work, we refer to particular occurrences of real-world events in a PAIS as *event instances*. We abstract from the technological details of how events are collected from different sources. Thereby, we work at the abstraction level of event logs that capture relevant

Event ID	Trace ID	Activity	Other attributes
e_1	1	Collect client data	-
e_2	1	Determine discount	VIP status = No, Age = 34[yrs]
e_3	2	Collect client data	-
e_4	2	Determine discount	VIP status = Yes, Age = 28[yrs]
e_5	1	Record discount	Discount = 15[%]
e_6	2	Record discount	Discount = 25[%]

Table 1: An example event log for the process of assigning a discount to a client

events of a business process grouped by the corresponding case, as is also defined in the process mining literature [184].

Below we specify definitions of event instances, attributes, traces and logs. When an event instance is recorded in a PAIS, among other attributes, it normally also has a unique identifier and a time stamp. We assume that an activity name in the process model corresponds to related event instance name in an event log.

Definition 17 (Event Instance, Attribute, Trace, Log). Let E=$\{e_1$, ...,$e_n\}$, $n \in \mathbb{N}^+$, be a finite set of n event instances and A=$\{A_1, ..., A_v\}$, $v \in \mathbb{N}^+$ a finite set of v attributes. Each attribute $a \in A$ is associated with a domain $Dom(a)$, which represents a set of either numeric, or nominal values. Each event instance $e \in E$ has tuples (a, y), $a \in A$, $y \in Dom(a)$ assigned to it. A trace is a finite sequence of event instances $e \in E$, such that each event instance appears in it once. An event log L is a multiset of traces over E. ◇

Note that in general event logs are not restricted to the components in this definition in general. Rather, many information systems keep track of additional aspects of events, e.g., roles, used resources, costs, data. However, in this thesis the data attributes, the values of which are assigned during process execution. We assume that the environment that generates the events also sets the timestamps of the events.

Table 1 shows an example event log for a process model of credit-risk assessment in a bank. Thereby, event instances are labeled by names of executed activities associated with values of data attributes attached to them. For each event instance e, the event log records the event instance ID, the trace ID referring it to the corresponding process instance, the name of the executed activity, and the subset of event attributes logged when a token is produced by the corresponding transition. For example, the event instance e_2 has the following attributes: $VIP\ status = No$, $Age = 34\ [yrs]$. All other information, e.g., the timestamps of event instances, is discarded.

2.4.2 *Executing Decisions Complementary to Processes*

During process execution, a point can be reached at which a decision needs to be made. If the principle of separation of process and decision logic is supported, the process activity that invokes a corresponding decision model supplies the decision-making system with input data. Next, the decision is "made", and the result is returned to the invoking activity.

Automating decisions in business processes involves identifying or modeling the business knowledge required to make those decisions and codifying it in a machine-executable form. The knowledge can then be automated by encapsulating it in services that execute the decisions. Next, decision management systems provide a means for executing decisions [70]. If decision models are represented by decision tables, such decision models are implemented and executed utilizing business rule engines [14, 54, 114, 117, 200]. A business rules engine is a software system which evaluates process execution data against a set of decision rules from decision table, and provides decision output. The matching of outputs against inputs is done by *inference engines*. Inference engines are provided by a variety of vendors, e.g., open-source project Drools, JRules, and Open-Rules, or commercial products such as Corticon. As well, there exist inference engines provided by scientific community, e.g., as in [4] and [198].

In our thesis, we abstract from the technological details of decision execution, which can be found in the works cited above. Thus, we only focus on modeling decisions complementary to process models. However, since implementing decisions is one of the goals of decision modeling, we assume that all the information necessary for decision execution is provided by decision models through the decision requirements layer, and the decision logic layer.

The integrated execution of processes and decisions can be illustrated as follows. When the process in Figure 13 is instantiated with the arrival of a new client, the bank worker retrieves *Client's record* data from the information system, and next, the worker performs activity *Collect client data* to get an information about the client such as *VIP status* and *Age*. The instantiation of decision activity *Determine discount* from the process model invokes corresponding decision from the decision requirements diagram depicted in the centre of the same figure. Decision *Determine discount* from the decision requirements diagram invokes decision table depicted in the right of the same figure. The decision rules in the decision table are evaluated against the client's data, and the inference mechanism yields the matching output discount. This data is used in the last process step by the worker during the execution of activity *Set discount*.

2.4.3 *Summary*

In this chapter, we discussed preliminaries for our work, main concepts of business process management and decision management domains. Firstly, we introduced the definitions of business process, business process management, and business process management system. Secondly, we explored which decisions are relevant for business process management and came to a conclusion that operational decisions are best suited for integrated process and decision management as they can be easily formalized, implemented and reused in processes. Considering the operation nature of decisions relevant for business process management, we introduced definitions of a decision, decision management, and decision management system. Further we explored state-of-the-art integrated usage of business process and decision management. Our conclusion states that although stakeholders need integrated solutions, they do not have a comprehensive methodology on how to design them, i.e. how to exactly realize separation of process and decision concerns introduced in [60, 142, 197].

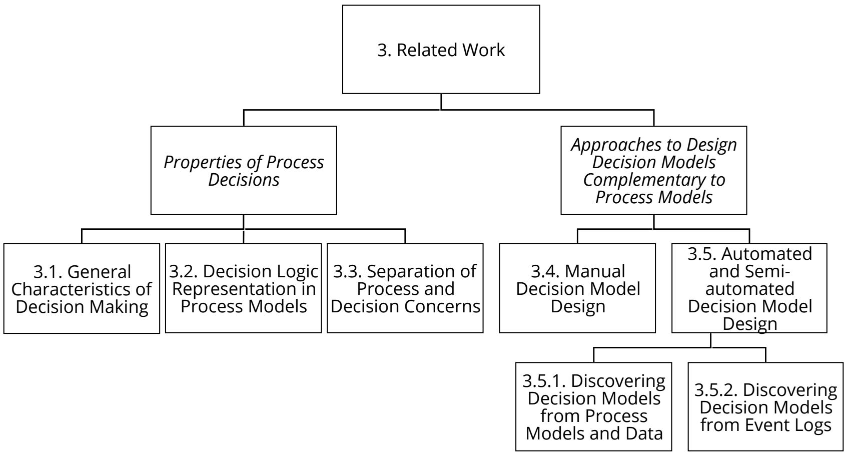
Next, we introduced integrated business process and decision management lifecycle which grasps such phases as design, configuration, enactment, and evaluation of decisions and processes. Although the phases of the introduced lifecycle are organized in a cyclical structure, the dependencies between phases do not imply a strict temporal execution ordering of them, especially if to take into account the the evolution of process and decision management at real enterprises is not necessarily simultaneous.

Afterwards, we provided formalizations of process and decision models. We motivated usage of the BPMN standard as a notation to describe processes as de facto standard, where the DMN standard is a logically complementary notation to describe process decisions provided already by many vendors. Using these notations, we introduced examples of process and decision models and demonstrated a possible implementation of the separation of concerns principle. The introduced formalization helped us to introduce later a formal specification of the elements and the relationships that form a decision model which can be designed and executed complementary to process model.

Lastly, we introduced formalisms connected to the execution of processes and decisions, such as process and decision instantiation, and event logs. This includes the notion of a process-aware information system that serves for management and execution of processes and decisions. The execution information about processes and decisions serves as an important source of information which can be included into the process of discovery of decision models complementary to process models.

3 RELATED WORK

In this chapter we elaborate on the work which is related to the research presented in our thesis. Figure 16 presents our literature classification. On a high level, the related works can be divided into two groups. The first group refers to the work on properties of decisions taken in processes. The second group is related to approaches to design of decision models complementary to process models.



3. Related Work
Properties of Process Decisions
Approaches to Design Decision Models Complementary to Process Models
3.1. General Characteristics of Decision Making
3.2. Decision Logic Representation in Process Models
3.3. Separation of Process and Decision Concerns
3.4. Manual Decision Model Design
3.5. Automated and Semi-automated Decision Model Design
3.5.1. Discovering Decision Models from Process Models and Data
3.5.2. Discovering Decision Models from Event Logs

Figure 16: Classification of related work

The first group is presented in Sections 3.1–3.3. We start with an exploration of general decision characteristics stemming from the decision theory in Section 3.1. We classify the decision characteristics into either those which are intrinsic to the decisions considered in our thesis (see Chapter 2), or those which could be considered in future works. In Section 3.2, we explore different approaches for representing decisions in process models and relate them to our work. Section 3.3 presents the related work which motivates and explains the paradigm of separation of process and decision concerns.

The second group is dedicated to different approaches to design decision models complementary to process models. Process decision design can either be made by humans, or automatically or semi-automatically derived from process-related information. We explore works dedicated to manual design of decisions in Section 3.4. The works dealing with automated and semi-automated design of decision models complementary to process models can also be divided into two groups, depending on which process information is available. Thus, Section 3.5.1 introduces existing techniques on discovery of decisions from process models. Next, Section 3.5.2 presents state-of-the-art approaches on discovery of decisions from event logs.

Next, Sections 3.1-3.3 present the works related to the process decisions properties, representation of process decision logic in process models, and the principle of separation of process and decision concerns.

3.1 GENERAL CHARACTERISTICS OF DECISION MAKING

Design of real-life decision models complementary to process models is a big challenge which implies the collaboration of business and IT professionals. Members of these communities are typically characterized by different professional backgrounds [202]. What could serve as a bridge between both sides, is the decision theory which is a discipline dealing with applications of mathematics, statistics, economics, management, and psychology for studying common factors influencing decision making and identifying the ways of its improving [66, 152, 174]. Further, we provide a classification of the general characteristics of decision making based on our literature review of the decision theory, with the aim to cover an intersection between the decision and business process management theories.

The core setting of decision theory is a decision problem that consists in an occurrence of a subject *decision maker* whose aim is to make an optimal choice between a finite set of *alternatives* resulting in a possible *outcome* event [152].

There exist multiple characteristics of the decision problem and its elements. Figure 17 presents general characteristics of the decision making derived by us from [33, 42, 66, 68, 131, 152, 207]. We divided these characteristics into decision factors and properties. *Factors* are used as categorical variables, and they are denoted with letter *F* followed by the factor number, e.g., *F1*. *Properties* express concrete characteristics of decisions, and they are denoted with letter *P* followed by the property number, e.g., *P1*.

As depicted in the figure, we distinguish 4 general factors influencing the decision problem: (1) Novelty; (2) Stucturedness of problem statement; (3) Information characteristics; and (4) Representation of elements of decision problem. As it can be seen from the figure, some of the factors are decomposed into subfactors. Herewith, we distinguish 33 decision properties and explore their relation to decision models designed complementary to process models, which are our research subject as defined in Section 2.3.3. The relation of general characteristics of DMN decision models, considered in this thesis, is presented in the corresponding column of Figure 17.

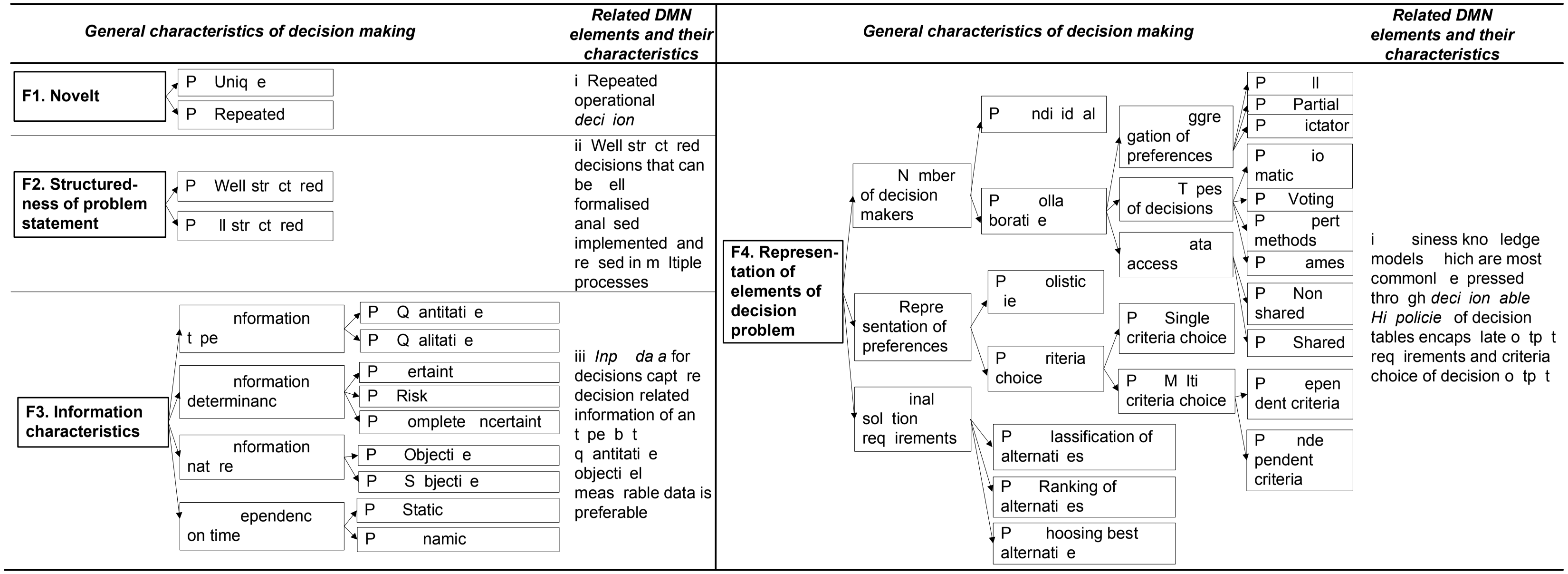

General characteristics of decision making
Related DMN elements and their characteristics
F1. Novelt
P Uniq e
P Repeated
i Repeated operational *deci ion*
F2. Structured-ness of problem statement
P Well str ct red
P Il str ct red
ii Well str ct red decisions that can be ell formalised anal sed implemented and re sed in m ltiple processes
F3. Information characteristics
nformation t pe
P Q antitati e
P Q alitati e
nformation determinanc
P ertaint
P Risk
P omplete ncertaint
nformation nat re
P Objecti e
P S bjecti e
ependenc on time
P Static
P namic
iii *Inp da a* for decisions capt re decision related information of an t pe b t q antitati e objecti el meas rable data is preferable
General characteristics of decision making
Related DMN elements and their characteristics
F4. Represen-tation of elements of decision problem
N mber of decision makers
P ndi id al
P olla borati e
ggre gation of preferences
P ll
P Partial
P ictator
T pes of decisions
P io matic
P Voting
P pert methods
ata access
P ames
P Non shared
P Shared
Repre sentation of preferences
P olistic ie
P riteria choice
P Single criteria choice
P M lti criteria choice
P epen dent criteria
P nde pendent criteria
inal sol tion req irements
P lassification of alternati es
P Ranking of alternati es
P hoosing best alternati e
i siness kno ledge models hich are most commonl e pressed thro gh *deci ion able Hi policie* of decision tables encaps late o tp t req irements and criteria choice of decision o tp t

Figure 17: Classification of general characteristics of decision making and corresponding DMN elements and their properties. Factors are used as categorical variables, and they are denoted with letter *F* followed by the factor number, e.g., *F1*. Properties express concrete characteristics of decisions, and they are denoted with letter *P* followed by the factor number, e.g., *P1*.

(F1) Novelty

With regards to *novelty* (factor *F1* from Figure 17), decisions can be distinguished as *unique* (property *P1*) and *repeated* (property *P2*), as discussed in [152]. In *unique* decisions, a problem or context are new for a decision maker. In such tasks, the decision maker dynamically identifies his preferences and makes a choice based on the analysis of the current situation. In contrast, in *repeated* decisions, a decision maker learns from previous experiences and applies some rules or procedures of the decision making which become available, as there is a possibility to observe results multiple times. With the course of time, the quality of such decisions becomes higher.

The unique decisions are not so well suitable for being modeled in processes, as the process model should serve as a blueprint for future executions of business process instances reflecting repeatable business situations [202]. In the DMN standard, the elements describing decision problems arising in processes are *decisions* (see Definition 4). As discussed in Section 2.1.2, DMN is rather aimed at supporting repeated operational decisions made in day-to-day business processes. Thus, the scope of properties of decision models considered in this thesis include only property *P2*, i.e. only repeated decisions are considered. However, some new approaches for complementary process and decision modeling also grasp the unique decision situations. For instance, in adaptive case management approach presented in [130], a case manager creates an individual execution path for each process instance (e.g., a doctor defining a clinical pathway for a specific patient). Thus, representation of novel decisions during design of decision models complementary to process models can be considered in future works.

(F2) Structuredness of Problem Statement

The decision problems can be divided into *well-structured* (property *P3*) and *ill-structured* (property *P4*) [174]. In management science, a problem is *well-structured* if following is true (cf. [174]):

- It can be described in terms of numerical variables, scalar and vector quantities;
- The goals to be attained can be specified in terms of a well-defined objective function, e.g., the maximization of profit or the minimization of cost;
- There exist algorithms that permit the solution to be found and stated in actual numerical terms.

A problem is *ill-structured* when it is not well-structured. An example of an ill-structured decision could be a credit-risk assessment decision taken by a bank worker based purely on his intuition.

The corresponding DMN elements encapsulating the structuredness characteristics are *decisions*. DMN prescribes management of decisions which can be well formalized, analyzed, implemented, and reused in multiple processes. As discussed in Section 2.3.2, for our work we focus only on the decision logic that can be represented with the help of decision tables. In such a manner, in our work we focus only on well-structured decisions. In other words, property P3 holds for our research subject, decision models designed complementary to process models.

(F3) Information Characteristics

Another factor influencing decision making in processes is *information characteristics* (factor *F3* from Figure 17), in conjunction with corresponding subfactors depicted in the figure.

The first subfactor is *information type* (factor *F3.1*). With respect to this subfactor, decisions are *quantitative* (property *P5*) if they are expressed in terms of numerical variables, or *qualitative* (property *P6*) if they are expressed in textual or verbal form. As in our thesis we consider decision models complementary to process models that are designed with the help of modeling notations, it can be stated that both properties *P5* and *P6* hold for them. Process modeling notations like BPMN support both types. For example, in BPMN the labelings of process activities express qualitative information, but the attributes values of process activities express quantitative information [138]. Analogously, in DMN the labelings of decisions and input data express qualitative information, but decision tables of the decision logic layer express quantitative and qualitative information. Although any type of information is acceptable, the preference is for quantitative objectively measurable data as one of the goals of decision and process model is future execution of models, which in such case will be less prone to errors due to possible misinterpretations of qualitative data.

The second subfactor, *information determinancy* (factor F3.2), refers to the predictability of the availability of information or data related to decision-making. Thus, decisions can be taken *under certainty* (property *P7*), *under risk* (property *P8*), or *under complete uncertainty* (property *P8*), according to [152]. In this thesis, we only consider decisions taken under certainty, since our goal is to provide decision models that can be executed complementary to process models in a deterministic way. The reason for this is that the decision models that we consider aim to capture repeated operational decisions which by nature are rather prescriptive and deterministic, i.e., those which satisfy property *P7*. However, future extensions of our work could include decision making under risk, and under complete uncertainty. For exapmle, incorporation of decisions considering all of the three properties mentioned above into BPMN models with the help of enhanced XOR-gateways was done in [40]. Future extension of our work could analogously include different types of information determinacy to DMN decision models designed complementary to BPMN process models, and provide corresponding discovery techniques.

With respect to the next subfactor, *information nature* (factor *F3.4*), decisions can be distinguished into two types. *Objective* decisions (property *P10*) are derived empirically from historical by some measurements or calculations. *Subjective* decisions (property *P11*) are normally human-made decisions. In BPMN, the information nature can be reflected at the modeling level, for example, through assigning the corresponding task types, e.g., human or service tasks (see Figure 6). That is reason why we consider that properties *P10* and *P11* are satisfied for decision models designed complementary to process models, the research subject of this thesis.

The final subfactor influencing a decision problem is its *dependency on time* which also allows to distinguish decisions into two types. In *static* (property *P12*) decisions, the information is considered as constant. If the information at the moment of decision making is a subject of change, such decisions are *dynamic* decisions (property *P13*). The static tasks are rare in reality and usually are considered as simplification of real dynamic life [152]. For example, the dynamic nature of decisions incorporated in process models (e.g., in conditions of XOR-gateways in BPMN process modes as demonstrated in [40]) can be observed when the process is instantiated. Then, process and decision execution operates on data about the current process instance which influences the decision outcome. From this point of view, the dynamism property is relevant to decision models that is our research subject, i.e., property *P13* holds for them.

(F4) Representation of Elements of Decision Problem

The last factor influencing decision making is *representation of elements of the decision problem* (factor *F4*), which naturally plays a significant role in decision formalization. Firstly, all decision problems can be categorized by a *number of decision makers* (factor *F4.1*) into *individual* decision making (property *P14*) with exactly one decision participant, and *collaborative* decision making (property *P15*) with more than one decision participant [33, 33, 42, 68, 131, 207]. In the case of collaborative decision problems, there exist a number of group-specific factors of decision making, e.g., the type of *aggregation of preferences* such as *full* (property *P23*), *partial* (property *P24*), or *dictator* (property *P25*) [152].

Another group-specific factor is *types of decisions* (factor *F4.1.2*), an example of which could be an aggregation of preferences in axiomatic type (property *P26*) according to a defined principle by taking mean or maximum of rankings. The rest of the decision types are shown in Figure 17 (properties P28-P30) and discussed in more details in [152]. Finally, in groups the decisions are made depending on the type of *data access* for participants to the information (factor *F4.1.3*), i.e. to the preferences of the other decision participants.

The decision processess can also be distinguished by the type of the representation of preferences. In *holistic* decision processes (P16), a decision maker is able to instantly form a comprehensive overview of the problem and make a decision based on his or her professional experience and intuition. For example, an experienced manager immediately knows which products will be in demand, to whom allocate a task execution. In contrast, in decision problems designed on top of *criteria choice* (property *P17*), it is supposed that a decision maker makes choices based on predefined *single* or *multiple criteria* (properties *P21* and *P22*). At last, the decision problems differ in *final solution requirements* (factor F4.3), which could be choosing classification of alternatives (property P18), ranking of alternatives (property *P19*), and choosing the best alternative (property *P20*).

In process models the decisions are represented by rather simple control flow structures as split gateways in BPMN, or by separating the decision modeling from the process model notation as in decision modeling notation [142]. Also, the decision nature of an activity can be represented by the business rule or activity type (see Figure 6). In the DMN standard, representation of elements of a decision problem can be done by both decision requirements level, and decision logic layer. Decision problem reflecting business know-how is captured in the DMN standard most commonly with the help of decision tables. Hit policies of decision tables encapsulate output requirements and criteria choice of decision output, i.e., properties *P17, P18, P19* and *P20* hold for DMN decision models that we consider in our thesis.

At the current moment, the resources involved in decision making are not graphically presented in DMN models: decisions can be only be indirectly annotated with information about involved decision makers or decision owners. Since in the current moment, the resource perspective is weakly supported by the DMN standard, we also exclude if from the scope of our thesis. In such a way, the decision making supported by decision models which we aim to discover from process models is viewed as individual, so that property *P14* holds. If the resource perspective gains momentum in further versions of the DMN standard, extension of our techniques for discovering decision models complementary to process models can be done. Also, at the present moment it is not possible to model complicated decision types in BPMN or DMN models.

3.2 REPRESENTATION OF DECISION LOGIC IN PROCESS MODELS

In this thesis, we investigate how decision models can be discovered from process-related information. As discussed by us in Section 1.1, this information can be discovered from process model and data. We present the concrete techniques to discover decision models from process models and data in Chapters 4 and 5. For positioning our work, below we present the related work dedicated to different ways of representing decisions in processes.

Typically, decisions are represented in process models as XOR-gateways [60, 63, 138, 202]. An example of expressing decision logic with the help of XOR-gateways is given for a discount assignment process in Figure 5. There also exist enhanced approaches on embedding extra-decision rules in process models. For example, in [40], the authors introduce the rule-based XOR-split gateway which allows to specify different kind of rules which route the process when making a choice which alternative path to follow after the gateway. Thereby, the traditional way of representing the decisions in BPMN with a control flow is enhanced, since the whole process fragments consisting of consequential activities are taken into account. The drawback of such modeling approach is that as soon as the decision logic becomes more complicated, it becomes very hard to embed it in a single XOR-split gateway.

On the other side of the related works spectrum are works dedicated to modeling decision logic of processes through rule-based constraints for process models. For example, in [153], a framework is proposed that is grounded in constraint logic programming for representing and reasoning about business processes from both the workflow and data perspective. One of the first attempts to provide a formal execution semantics for expressive BPMN workflows in the presence of data and arithmetic constraints is presented in [96].

Our work presented in [18] was one of the first to consistently analyze a process model repository consisting of around 1000 real process models, and to come up with patterns of embedding decision logic in control flow. As we discovered, indeed the majority of process decisions are encoded in process models through fragments consisting of XOR-gateways and corresponding control flow of activities. The results of our analysis cound be found in Chapter 4.

Another way to represent decisions in processes is through data flow of process models. Our works [19, 20] explore how decisions can be modeled in process models through data-flow structures. In our works we show that decisions can be encoded in attributes of data objects. Among other data-centric approaches which can be used for modeling process decisions is a methodology of *Product-Based Workflow Design* defined by means of a *Product Data Model (PDM)* [158, 180, 189]. In this approach, components of a decision outcome (product) are defined by data elements representing the smallest, meaningfully distinguishable portions of information. PDM consists of a finite set of data elements, data processing steps, and corresponding rules applied on top of them. There exist other data-centric process modeling approaches that use different types of data representation and input, such as the ontology-based knowledge-intensive approach in [154], or an approach on enhancing DMN and declarative process models [127]. The above mentioned approaches can be useful to model process decisions, but only in the cases where the process is purely data-centric. However, the focus on data often downplays the holistic view that should be achieved to support process decisions. In our work, we adhere to the principle of separation of concerns where decisions are modeled to complementary to processes, which allows for activity-centric focus in process models, and data-centric focus in decision models.

Another way to integrate decision logic into process model is through annotations for activities [199]. In this case, the textual elements are attached to graphical symbols within the process model to represent extra information about process decision making. For example, the routing rule for discount assignment from Figure 5 "If the client is VIP, and he is older than 40 years old, then assign 29% discount" can be simply annotated as a textual commentary for a process model. The works on natural text processing [73, 110] can be used to analyse such annotations and discover decision models from such textual commentaries, but this is out of the scope of this thesis.

To sum up, in existing approaches the process decision logic is expressed mostly through control-flow structures containing XOR-gateways. In our work we consider both control flow and data perspectives. We provide a methodology to extract decision logic from control flow of process models in Section 4, and from data flow of process models in Section 5.

3.3 SEPARATION OF PROCESS AND DECISION CONCERNS

As discussed in Chapter 1.1, we use the principle of separation of concerns as a motivation for our research on discovery of decision models complementary to process models. Below we introduce the related works which are dedicated to the separation of concerns principle, and link it to our work.

The separation of concerns principle has attracted a lot of attention in the domains of software modeling and design [60, 80, 102, 197]. It was pointed out in [179] that managing decisions separately from processes is important for agility and traceability of both. According to [158], if decisions are not explicitly designed in a dedicated model, that could lead to problems during execution of decisions. In [180], it is also pointed that modeling languages such as BPMN are not meant to represent the complex decision logic, and that inclusion of detailed decision logic in process models often results in complex, spaghetti-like models.

The concrete exploration of how to model decision logic complementary to process logic was done in [76]. In this paper, the authors introduce several kinds of business rules that can be used to generate less complex contol-flow-based process models. They include deontic assignments that specify data access for process and decision participants, and reaction rules describing actions that are to be taken as results of events. Debevoise et al. in [60] distinguish several categories of operational decisions that can be included in a decision model designed complementary to process model. The examples include inclusive task sequencing, participant assignment, exclusive gateway, and data coming from events preceding decision tasks. The externalization of business logic into business rules implemented as a process activity type was also discussed in [75].

The works mentioned above argue for separation of concerns, and propose several patterns of inclusion of decision-related process information into decision models. However, a holistic methodology on extracting such kind of information from process model is lacking. Therefore, we argue that there is space for extensive research and development in the area of mining decision models complementary to process models.

In [77], the authors state that *declarative* modeling approaches lead to more design- and run-time flexibility, better compliance guarantees, and higher expressibility. Since decision models are declarative in nature [197], combining them with business process models could help to achieve these benefits, and additionally preserve the benefits of separation of concerns. However, adaptation decision rules in such cases would be limited to handling changes in values of the data model, i.e. the adaptation can only manipulate the data model values and not the process itself. If a change in the process itself is required, some extra specification is needed.

Although one motivation for separation of concerns is the potential to increase flexibility of process and decision models, in this thesis we do not consider how to evaluate this characteristic of models. Nevertheless, there exist a plenty of related works that can be of use for achieving this. For example, Reichert et al. [156] introduce flexibility measures for processes which can be used in future works for evaluation of how flexible do process model become if their decision logic is externalized in a complementary decision model. An analysis of flexibility of processes enhanced with such decision formalisms as decision tables is addressed in [31] which demonstrates the agility benefit of decision-aware processes.

In [105], the authors introduce an approach for managing business process variability based on *configurable* process modeling notation incorporating features for capturing resources, data and physical objects involved in the performance of tasks. In configurable process models, the variations of processes are capturing mostly process properties, whereas in separated decision and process design, the decision models capture mostly declarative decision logic. However, decision modeling can be potentially applied complementary to configurable process models.

Although the separation of concerns principle has gained popularity among both academic, and industrial circles, it is important to note that there exist situations under which it is better to model a decisions as a part of a business process model. Thus, Wang et al. in [199] distinguished factors which influence the question of integration of business rules in process models. The top factors "for" separation of concerns are the ones that we also use in Section 1.1 for motivation of complementary usage of decision and process models: increased agility, rate of change, reusability, and accessibility of decisions in processes. However, the authors prove that there are two situations when it is better to integrate it in a business process model: (1) When a decision rule changes infrequently; (2) When a rule's reusability is low. Thus, if in a real process, any of these two situations can be observed, we recommend to exclude corresponding decisions or decision rules from being considered as a part of a decision model to be constructed complementary to process model.

Various Platforms and Ontologies

Lately, both scientific and business communities shared increasing interest towards exploring the decision support in organizations demonstrated by an increasing number of emerging approaches on different decision ontologies [104] and decision service platforms [6, 210].

The industrial solutions for separation of decision from process logic include SAP Decision Service Management [6], IBM Decision Manager [29], or Signavio Decision Manager [172].

In [102], a tool chain for creation of both models relying on concrete infrastructure and business rules is presented. The authors of [104] present a decision ontology for supporting decision-making in information systems. Another decision ontology is proposed in [30] as a "domain-specific modeling method that is integrated with an existing enterprise modeling method for describing and communicating decision processes".

A promising approach on viewing decisions "as-a-service" complementary to application process logic is discussed by Zarghami et al. [210]. Their approach introduces both synchronous interaction between process and decision service, and asynchronous interaction. Synchronous request-response manner means that the process always call the decision service at some predefined decision points, which is supported by the DMN notation. Zarghami et al. introduce asynchronous interaction such as that there is a change in the environment, the decision service notifies the process at runtime, thereby adapting it. Such type of communication could be taken into account by modeling decisions, but we do not do it as we stick to the DMN standard.

In order to deal with the changes that can arise from runtime contextual changes or the change of user requirements and preferences, an approach for decision services modeling is proposed in [197]. Thereby, the authors propose that the "stable" process logic is specified in terms of processes while rules are employed to specify conditions and constraints which can be dynamically adapted according to runtime circumstances. The rules are exposed as an independent decision service which can be called from the process.

Despite an increased interest towards complementary process and decision modeling, the stakeholders are still missing a holistic approach to automatically externalize decision logic of BPMN process models into a complementary DMN decision model, which we aim to cover in our thesis.

DMN and other OMG Standards

There exist a number of the OMG standards aiding design, execution and maintenance of enterprise decisions and processes. Below we discuss the related OMG standards, an overview of which can be seen in Table 2 .

The OMG consortium mandates separation of process and decision concerns. Thus, it recommends to model processes with the help of the BPMN notation [138], and to model decisions complementary to processes with the help of the DMN notation [142]. Thus, in this thesis we decided to consider the DMN standard for exploiting the decision logic alongside BPMN for exploiting the process logic. The DMN standard has been designed to work alongside BPMN, providing a mechanism for modeling the decision-making represented in a task within a process model. It is not obligatory to use DMN in conjunction with BPMN, but these two standards are highly compatible. Works on separation of process and decision concerns - [197], [60] and [102] - utilize these standards but do not provide a methodology of how to separate concerns. In [60], the authors provide the concrete examples of BPMN models with the integrated elements of DMN.

Another OMG standard, namely Case Management Model and Notation (CMMN) [141], exists for modeling cases of business processes, which are used to represent less structured or discretional activities. Similarly to decision activities in BPMN, the CMNN cases can contain decision tasks which can invoke corresponding DMN decision models. Thereby, the CMMN standard indicates that inputs of

Standard	Decision-related artefacts	Decision-related functions
BPMN [138]	Process model	-Description of decision-making steps in process models; - Invocation of a DMN decision model
CMMN [141]	Case model	-Description of decision-making steps in case models; - Invocation of a DMN decision model
DMN [142]	- Decision requirements diagram; - Decision logic	- Description of decision requirements and logic; - Provision of decision outcome data to a BPMN process model
SVBR [143]	- Business rules; - Business vocabularies	- Defines business vocabularies and facts for a DMN decision model
BMM [139]	- Business objectives	- Provides strategic guidance for modeling process and decision models

Table 2: An overview of the OMG standards related to decision making in processes

a CMNN decision task can be mapped to inputs of the corresponding DMN decision model, and outputs of the DMN decision model can be mapped to outputs of the CMMN decision task [141]. The guidelines on modeling decisions complementary to case models were provided, for example, in [97] and [84]. Although separation of concerns for case models and decision models is not addressed in our thesis, discovery of decision logic from case models or context information appears to be an interesting research question for future works.

The OMG consortium also presents another standard for representing decision rules in processes: Semantics of Business Vocabulary and Business Rules (SBVR) standard [143] which provides tools for specifying business rules and vocabularies in natural text expressions. This notations is mostly targeting the people who do not have substantial technical knowledge, but have a deep knowledge how the company works. In [101], the translation from SVBR to BPMN and

DMN models is presented, which could aid stakeholders already having SVBR rules designed in order to utilize the DMN models. Therefore, the translation mentioned above can be used complementary to techniques presented in this thesis, in case if a company does not have BPMN process models or execution data, but it has SVBR models.

The Business Motivation Model (BMM) [139] can be used for strategical guidance for designing and executing the DMN decision models. In particular, the business objectives defined by the BMM model can be associated to business performance attached to decisions in decision requirements diagrams. This allows to measure and monitor the related decision model's effectiveness, which can potentially serve for improving decision making in processes.

In Sections 3.1-3.3 we discussed the works related to properties of process decisions. Sections 3.4-3.5 present works from the second group of related literature which is related to discovery of decisions from process-related information. Section 3.4 discusses works on manual design of process decisions and are they related to research conducted in this thesis. Further, Section 3.5 presents related works on automated and semi-automated discovery of decisions from process-related information. Afterwards, the chapter is concluded by a short summary of all the sections.

3.4 MANUAL DECISION MODEL DESIGN

Process decisions can be either designed by humans, or automatically derived from process-related information. Our thesis proposes semi-automated techniques for discovery of decision models from process-related information such as process models or event logs. However, manual design of decisions complementary to process models can be viewed as an alternative way for modeling decisions in processes. Below we provide an overview of different methodologies for manual design of process decisions, and propose when they can be used instead of our approach for discovery of decision models from process-related information. Since there exist multiple techniques for modeling decisions, we consider them in the section below in a historical order of appearance, as presented in Figure 18.

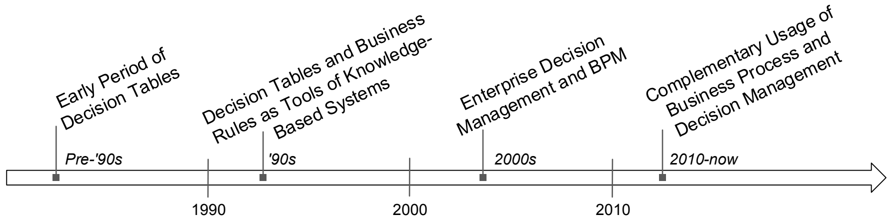

Early Period of Decision Tables
Decision Tables and Business Rules as Tools of Knowledge-Based Systems
Enterprise Decision Management and BPM
Complementary Usage of Business Process and Decision Management
Pre-'90s
'90s
2000s
2010-now
1990
2000
2010

Figure 18: Decision modeling timeline

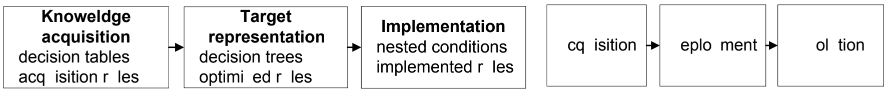
Pre-'90s: Early Period of Decision Tables

The history of operational decision management finds its origin in decision table modeling, whereby rules used for decision making are represented in a set of related decision tables. Earlier descriptions of the decision table concept can be found in [121, 148]. The Canadian Standards Association (1970) was the first to issue a formal standard for decision table modeling and execution. The ability to represent conditional logical expressions in a compact manner and the ease of adaptation and consistency checking were considered as a solution to the problem of growing software complexity [45]. Originally, the decision tables were manually translated into code. A number of specific languages and initial preprocessors were developed in '60s in order to convert two-dimensional decision tables into linear program code. Later, an attention was paid to optimization of the conversion process, followed by emanation of more powerful commercial preprocessors [150, 196]. The experiences and best practices on decision table modeling and execution [45] concluded that, when properly used, decision tables can serve as an excellent tool for problem analysis, specification, and implementation. Starting from '80s, the application area of decision tables was extended from programming towards various other domains such as knowledge engineering and validation, conditional logic representation, rules and regulations, etc. [190]. Furthermore, the emphasis moved towards the power of decision tables to represent complex decision situations in a simple manner, allowing non-complicated checks of their consistency, completeness and correctness [49, 135].

'90s: Decision Tables and Business Rules as Tools of Knowledge-Based Systems

In the '90s, the role of decision-support tools for companies play knowledge-based systems [192], the lifecycle of which is represented by a set of phases of knowledge acquisition, target representation, and implementation (see Figure 19a). During the acquisition phase, business know-how is acquired and verified. Next, this know-how is transformed into a suitable representation, depending on the problem characteristics. Finally, the knowledge formalisms are implemented, considering such factors as order and relevance of the rules and conditions, execution time efficiency etc.

In parallel to the evolution of knowledge-based systems, another view on decision-making support is developing in this period which is represented by a concept of business rules management [161]. The phases of business-rules management lifecycle are similar to the phases of knowledge development lifecycle and they are presented in Figure 19b. During the phase of acquisition of business



(a) Knowledge development lifecycle (cf.[192])

(b) Business-rules lifecycle (cf. [161])

Figure 19: Different views on decision-making phases in '90s

rules from Figure 19b, enterprise objectives and constraints are collected for generation operational business rules. During the deployment phase, the rules are realized by an enterprise information system. In the evolutionary phase, based on the information provided by the monitored data, it is determined if some business rules are incorrect or incomplete, and corresponding improvements are conducted. From Figure 19, it can be seen that the phases of business-rules management are similar to the phases of decision table management, and so are these two concepts, although during the discussed period they are represented by different streams of works.

During this period, decision tables serve as a powerful tool for implementing the knowledge development lifecycle [191]. The decision table management lifecycle of this period often complies to the phases presented in Figure 20. It starts with identification of the lists of business conditions, their states, and possible actions. During the phase of decision rule definition, a decision problem is described as a series of logical "if-then" relations which map combinations of condition states and actions to be executed. Once the input data is defined, the decision table is constructed, and further, it is checked for completeness, correctness and consistency. Simplification of decision tables happens if it is possible to contract tables by minimizing the number of their columns or rows for the given condition order. At last, optimization of decision tables can be done, e.g., by techniques of optimal decision tree generation [112, 123].

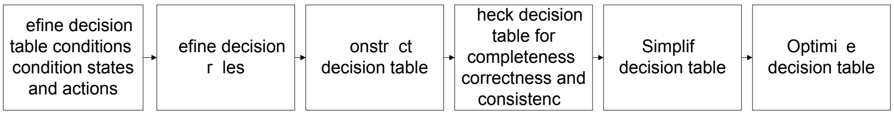

efine decision table conditions condition states and actions
efine decision r les
onstr ct decision table
heck decision table for completeness correctness and consistenc
Simplif decision table
Optimi e decision table

Figure 20: Stages of decision table management in the '90s (cf. [191])

2000s: Enterprise Decision Management and BPM

Around 2005, a new concept of Enterprise Decision Management (EDM) appears, which promotes flexible information-driven processes and conversion of data into intelligence which is able to guide and execute enterprise decisions [124]. The EDM approach implies application of rule-based systems in conjunction with analytic models in order to automate, improve, and distribute decision-making capabilities across an organization [82]. The newly defined knowledge management lifecycle typically enables shorter change cycles, and it shifts more responsibility for stewarding business changes from technical to business people. Around this time, agility was identified as a key criteria for successful decision automation, whereby business rules management systems (BRMS) were identified as a primary mechanism for automating decisions [176].

In parallel to the evolution of decision management, starting from the early '90s, a concept of business process management is steadily developing as an independent discipline including concepts, methods, and techniques to support the design, administration, configuration, enactment, and analysis of business processes [202]. Eventually, it was understood that information systems often make business processes more rigid than flexible. For example, the companies struggled to promptly guarantee compliance of business processes and different kind of regulations. With that, the flexibility of information systems dictated by constantly changing customer and market requirements was acknowledged as a determining factor for a company [171]. The work of [76] was one of the first suggesting to use business rules in business process modeling which allowed better traceability of the execution of business policies and business protocols. However, the main shift towards the concept of integrated business process and decision management was observed in the next decade.

2010–now: Complementary Usage of Business Process and Decision Management

In recent years, a number of industrial decision service platforms have appeared in addition to existing BPM-systems, e.g., SAP Decision Service Management [6], IBM Operational Decision Management [48]. On the academia side, a number of works dedicated to conceptual decision service platforms [30, 210] and decision ontologies [104] complementary to business process management techniques have been proposed. Although it is evident that stakeholders need integrated solutions on business process and decision management, a paradigm of separation of process and decision concerns is also gaining weight [94, 197]. It has been observed in [18] that business process models are often misused for modeling decision logic within the control and data flow of process models. Because of such hard-coding of decisions in process models, organizations often lack necessary flexibility, maintainability and traceability in their business operations [177]. To create a standard approach for describing and modeling repeatable decisions within organizations and ensure that decision models are interchangeable across organizations, the Decision Model and Notation (DMN) standard was developed by the OMG group [142] . The DMN standard is aimed to be complementary to the BPMN standard [138].

The DMN standard appeared recently, and there exist only mostly industrial-oriented guidelines, e.g., as the ones provided by Debevoise et al. in [60]. Nevertheless, there exist works prescribing design of decisions which can be reused for modeling DMN decision models. Thus, in [15], standard steps of decision making are presented, such as defining the problem, identifying problem parameters, an selecting method for decision making. In [208], Feng et al. propose a more specific approach, *Decision Dependency Design (D3)*, which is a methodology for designing process decisions and their dependencies, abstracting from detailed decision logic. The introduced notation is somewhat similar to the DMN decision requirements diagram, but the visualization of notations' elements differ, and the information requirements for decisions are more enhanced in *D3* than in the DMN standard. The authors describe the major phases that can be distinguished for creating corresponding decisions model such as gathering requirements, eliciting input dependencies and constraints.

A similar approach for modeling DMN decision requirements diagram is the *Decision Requirements Analysis Workshop (DRAW)* technique presented by A. Fish in [70]. DRAW is a structured workshop technique allowing a methodical top-down analysis of a decision into the structure recorded in the DRD, and it can be fully applied for modeling DMN decision requirements diagrams. An approach for modeling the DMN decision logic layer is described for decision tables in [160] through construction of a decision-goal tree partitioning a decision into subdecisions.

All of the existing methodologies of manual design of decisions models presented in the section above can be used independently or complementary to our methodologies of semi-automated discovery of decision models from event logs. We provide an overview of selected works on manual design of decision models in Table 3 (rows $1-3$), where we compare them with works on semi-automated decision model design discussed in the next section (rows 4-8). The manual decision model design can be used, when there exist neither process models, nor event logs describing business process and its behavior. In such cases, it is recommended that process or decision analysts apply the methodologies discussed above to model process and decision models "from scratch".

3.5 AUTOMATED AND SEMI-AUTOMATED DECISION MODEL DESIGN

While decision models can be created by experts, manual design of a decision model complementary to process model can be a difficult and time-consuming endeavor. In order to construct correct and reusable process and decision models, one must be wellacquainted with the many elementary processing steps and their interrelations [180]. Given the right tools, automating the process of decision model design can yield major benefits, such as improved quality of decision models, their improved explanatory power, increased consistency with the process, reduced human involvement, etc. Striving to increase the automation rate for the process of extracting decision models from process-related information, we developed a set of corresponding techniques that constitute the core contribution of this thesis presented in Part II.

The process of extraction of decision logic from process-related information is also often called "decision mining" [147, 165]. Analogously to this, we refer to the process of discovery of decision models complementary to process models as "decision model mining". The difference between these two terms consists in the fact that decision models consider not only stand-alone process decisions, but additionally, decision dependencies and their requirements (see Definitions 12 and 13). Below we present related work for mining of decision models from process models and data, event logs, and other process-related information.

3.5.1 *Discovering Decision Models from Process Models and Data*

Although it is not considered good practice to model the detailed decision paths in the business process model [197], there have been a few works which propose concrete methods of externalizing decision logic taking process models or data as inputs. An overview of these works can be found in Table 3. Rows $1-3$ refer to works discussed in Sections 3.4, and Rows $4-8$ refer to works discussed below in this section.

The presented table firstly presents the selected works from the previous section dedicated to manual design of decision models. It can be seen from the table, that there exist just a few works on semi-automated design of decision models complementary to process models, in comparison to works dedicated on manual design of such models.

There exist a few academic works proposing their own decision-making ontology to be used complementary to processes, e.g., by Kornyshova et al. in [104], or by Bock et al. in [30] (see Row 4 of the table). Both of this works derive elements intrinsic to decision processes and create independent decision-making ontologies. At the same time, these two works are more conceptual and do not provide means for automated discovery of the decision-making ontologies. However, these ontologies contain a richer set of decision-related elements, such as decision stimulus or goals, than DMN decision models used in our thesis. Although we see the work presented in this thesis as a basis for discovery of decision models complementary to process

Row	Authors	Input		Output		Decision Design
		Business Know-How	Process Model	Decision Model	Improving Decision Making	
1	Vanthienen et al. [191]	+	-	- (Decision table)	-	Manual
2	Feng et al. [208]	+	-	D3	-	Manual
3	Roover et al. [160] A. Fish [70] Debevoise et al. [60] Taylor et al. [178]	+	-	DMN	-	Manual
4	Kornyshova et al. [104] Bock et al. [30]	+	-	Author's ontology	-	Manual
5	Ghattas et al. [74] Wetzstein et al. [203]	+	+	-	+	Manual
6	Van der Aa et al. [180]	+	- (Process data)	DMN	-	Semi-automated
7	Batoulis et al. [18]	+	+	DMN	-	Semi-automated
8	Bazhenova et al. [20]	+	Part of process model	+	-	Semi-automated

Table 3: Selected works related to decision models extraction from business know-how, or from process models (discussed in Sections 3.4 and 3.5.1)

models on example of DMN decision models, interested stakeholders can utilize our approach and enrich DMN decision models with decision elements presented by the mentioned ontologies. This would require an appropriate mapping of corresponding decision elements.

There also exist a number of works which deal with improving business process decision making based on past process executions, e.g., [74, 203] (see Row 5 of the table). Since we are focused on extraction of decision models complementary to process models, improvement of process and decision execution is not considered in our thesis. Nevertheless, dynamic improving of decision making can be viewed as a next step which can be applied to decision models once they are discovered complementary to process models.

In [180], the authors propose a methodology for derivation of DMN models complementary to BPMN process models from *Product Data Models (PDM)* which lie at the core of the product-based workflow design concept (see Row 6 of the table). This work considers only data-centric processes, whereas in our work we aim at discovering decision-related information from process models like BPMN models which contain both data and control flow. However, this work can be recommended to use for discovery of decision models complementary to process models, if an interested stakeholder has no process model at their exposure, but only a product data model or an analogue of it.

Process models can also be discovered or generated from decision models. In [35] and [208], the authors decompose decision structures into goal graphs that are finally used to design a process model. The advantage of such approach is that the process models generated for decision models can provide an enhanced execution mechanism for decisions, since a certain control flow for decision execution is prescribed. However, execution of decision models goes beyond our problem statement formulated in Chapter 1, as we are only interested in the design phase of business process management and decision lifecycle (see Section 2.2). With that, decision model execution refers to the second phase of the lifecycle. However, it seems perspective to consider the questions of generation of process models from decision models in future works. The existing works mentioned above can be recommended to stakeholders that have no process models, but only decision models at their hands.

In general, it can be observed that there exists a research gap on elaborated methodologies on extraction of decision models from process models and data. This thesis is aiming to overcome this gap. Our techniques for discovery of decision models from process models and data are presented in Chapters 4 and 5 correspondingly. They are based on our works by Batoulis et al. [18] (see Row 7) and by Bazhenova et al. [20] (See Row 8). The distinguishing difference of our techniques from existing works is that they extract decision models taking process models as an input. Thereby, the decision logic can be externalized from the process model in a semi-automated way. As an output, our approaches produce decision models that can be executed complementary to process models, which realizes the separation of concerns principle.

3.5.2 *Discovering Decision Models from Event Logs*

Decision-making in business processes typically results in a vast amount of data, which can be analyzed afterwards according to certain parameters suitable for assessing the enterprises goals. To provide such kind of analysis, in Section 6 we propose a methodology for deriving decision models complementary to process models from event logs. More specifically, we adapted a widely applied decision tree mining algorithm, C4.5 (see, for example, [205]), for solving the machine learning problem of discovering of decisions from process event log, and their dependencies. The methodology is based on our publications [21] and [23]. We also provide an extension of the methodology with fuzziness based on our work [24].

An overview of the related works on mining decision models from event logs are presented in Table 4.

There exist multiple algorithms for discovery of the decision rules, among them three approaches are most widely considered in the literature [12, 116, 166]. The first one – solving the decision mining by constructing a *neural network* – gives very accurate classifications, but the output rules might be quite complex which is then hard to translate into a self-explanatory decision model [12]. The second one – mining the decision rules with the help of *support vector machines* – also provide very accurate classifications, however, there are no existing solutions of translating the output support vector machines into the business rules [116]. With that, the application of the third approach (see Section 6) – solving the rule classification problem with the help of the decision tree classification – allows easy creation of business rules from the output decision trees, although the classification accuracy is worse in this case than in two approaches mentioned above [166]. The authors of [120] also state that the output of decision tree mining can easily be interpreted by humans and the extracted knowledge can be clearly presented and visualized. That is why for our algorithm for discovering decision models from event logs we chose a variation of decision tree mining, C4.5 algorithm, as a basis. The results of applying of our approach for discovering decision models from event logs highly depend on the input data. Investigations on sensitivity of the approach to different data inputs for the algorithm used in our work - decision tree mining - can be found in [9].

Row	Authors	Output Decision Models	Consideration by the Decision Discovery of:				
			Control Flow Decisions	Data Decisions	Decision Dependencies	KPIs	Fuzziness
1	Baesens et al. [12] Lovell et al. [116] Ludwig et al. [120]	-	-	+	-	-	-
2	Rozinat et al. [165]	- (Decision points)	+	-	-	-	-
3	Dunkl et al. [64]	- (Decision points)	+	+	-	-	-
4	De Leoni et al. [53] Mannhardt et al. [122]	- (Decision points)	+	-	-	-	-
5	Polpinij et al. [149]	-	+	-	-	-	-
6	Batoulis et al. [17]	DMN decision model	-	+	-	-	-
7	Lakshmanan et al. [107] Petrusel et al. [147]	Markov decision process	-	+	-	-	-
8	Becker et al. [26]	Probabilistic finite automata	-	+	-	-	-
9	Petrusel et al. [146]	*Decision process* model	+	+	+	-	-
10	De Smedt et al. [58]	DMN decision requirements diagram	+	+	+	-	-
11	Bazhenova et al. [23]	DMN decision model	+	+	+	-	-
12	Bazhenova et al. [21]	DMN decision model	+	+	+	+	-
13	Bazhenova et al. [24]	DMN decision model	+	+	+	-	+

Table 4: Selected works related to decision models extraction from event logs (discussed in Section 3.5.2)

The work of Rozinat et al. [165] describes the extraction of decision rules for control flow decisions from event logs. In comparison to this work, we additionally identify data decisions, and decision dependencies. [53] extends [165], but in contrast to our paper, the authors seek to improve the performance of the rule extraction algorithm for control flow decisions, while we seek to complete the decision knowledge derived from event logs beyond control flow decisions. Another extension of [165] was done by Dunkl et al. in [64] by augmenting the space of possible discovered decisions with attributes obtained from event log.

Mining business rules from business process repositories was also done in [149], but they focus on only activity execution logs, without taking data flow at all into account, and also they do not consider dependencies between sets of rules.

Mining of complex decision logic for process branching conditions with the help of decision tree learning was done by Mannhardt et al. in [122]. In particular, the authors put forward a general technique to discover branching conditions where the atoms are linear equations or inequalities involving multiple variables and arithmetic operators. In contrast to our work, this work does not mine dependencies between process decisions, but rather focuses on discovering of decision logic for standalone decision points. However, they are able to discover more complicated decision rules, than are adapted C4.5 algorithm. Therefore, future works can consider investigation of using their more algorithms for decision logic discovering in conjunction with our holistic approach to discover decision models and their dependencies.

In [17], an approach to construct decision models from historic process execution data is presented. It can be used also for solving our problem, but comparison to our approach from Section 6, it utilizes additional steps of derivation of a Bayesian network, followed by creation of a corresponding influence diagram which is further transformed into a DMN model. In our approach, we avoid these intermediate steps by applying C4.5 classification directly on data, directly obtaining DMN model. Also, this approach only takes into account dependencies between data attributes in the event log. However, we also incorporate discovery of control flow decisions into an output DMN model, thereby making the output decision model more fully explaining the process decisions.

Among other approaches for mining decisions from process data is the one described in [107] where instance-specific probabilistic process models and their translation to Markov chains enable predictions about future tasks. Similarly, in [26], an approach based on probabilistic finite automata to predict future events in running process instances is developed. However, both works do not derive decisions from obtained nodes and dependencies between them. In [147] the authors derive a probabilistic Markov decision process from event logs. However, the authors look at business decision making only as at a data-centric environment. In contrast, in our work we derive decision models from both control and data flow execution data recorded in event logs.

Another approach on mining decisions from event logs is presented in [146]. The approach is different from ours as it yields as its outcome not really a decision model, but rather a decision process which incorporates some elements of control flow like XOR-gateways. Such approach might be helpful for decision-intense processes where the whole process is about a decision. However, as we discuss in Section 2.3, one of the principles that we pursue is the separation of concerns principles, according to which process models should be imperative, and complementary decision model should be declarative.

In the recent work by De Smedt et al. [59], the authors propose a framework to position the existing works related to decision discovery, in the context of business processes. The authors introduce the notion of maturity of decision models which are designed complementary to process models. According to the authors, the most mature decision model incorporates both the data recorded in the event log that captures the process execution and the dynamic behavior of the activities from the corresponding process model.

In the further work of De Smedt et al. [58], the first step was done to approach such level of maturity for decision models discovered from event logs. Taking an event log as input, the approach discovers a decision requirements diagram that incorporates information about the process model elements that carry a decisional value. In particular, the presented approach focuses on different types of activities that are present in a process model discovered from the event log, and establishes how they contribute to the corresponding decision model. The work of De Smedt et al. considers process activities as the drivers of decisions, whereas we assume that the decision drivers are control flow decision points and data attributes, as presented in details in Chapter 6.

Correspondingly, the dependencies in the output decision requirements diagrams that are derived with the help of the approach presented in [58], reflect the connections between the decision-related activities and data of processes. However, the dependencies in the output decision requirements diagrams that are discovered by applying our methodology from Chapter 6, reflect the connections between control flow decisions and data of process models that carry a decisional value. Therefore, in comparison to the approach from [58], our methodology to derive decision models outputs the decision requirements diagrams of a different nature. It can be recommended for stakeholders to use the approach from [58] for the situations where activities of different types that carry decisional value are present in an input event log. In the situations where rather control flow and data attributes carry the decisional value, it would be recommended to use our approach, which is presented in Chapter 6.

Enhancement of Decision Model Mining with Fuzziness

Commonly, decision logic is represented by rules built on top of Boolean algebra. Since real-life decisions often deal with imprecision, the formal nature of decisions based on Boolean algebra is often hard for interpretation und utilization in practice [28, 38, 81, 209]. Operations research considers fuzzy logic, based on fuzzy algebra [209], as a tool dealing with partial knowledge. In Chapter 7, we propose an approach to incorporate fuzziness into DMN decision models. Furthermore, we propose a methodology for discovering fuzzy DMN decision models from event logs.

As business process modeling often describes uncertain and error-prone behavior of humans, incorporation of fuzziness is helpful as shown in [28]. The authors demonstrate that introducing fuzzy logic in processes has following benefits: (1) it allows inaccurate data in process executions; (2) effectively incorporates with assigning to variables linguistic terms coming from human such as "little", "some", or "a few". By choosing appropriate rules, the process modeler can incorporate in the model data coming from subjective sources "without compromising the validity of the model".

Initially, fuzzy logic was widely applied in control theory, systems analysis, and artificial intelligence [100]. Eventually, fuzzy logic was also considered in business processes as a method for evaluating of judgements of decision makers in business processes with the help of fuzzy analytical hierarchy process in [164]. However, the method is aimed rather at solving constraints problem in the presence of multiple decision makers, and is not considered from processes or decision notation points of view.

Incorporation of fuzziness within single decision tables was done by Vanthienen et al. in [193]. Consideration of fuzziness in processes is presented in [47] where the authors extend the BPMN notation with fuzziness for modeling crime analysis processes being able to capture both the vague nature of forensic data and the uncertainties and conjectures characterizing the inference structures of this domain. In this work, the authors extend the BPMN elements like data objects and messages with fuzzy attributes and fuzzy constraints. In contrast to two works mentioned above, our approach additionally allows for designing of fuzzy decisions and their dependencies within a dedicated decision model.

A methodology aimed at making fuzzy decisions again in the context of crime business processes is introduced in [7]. The presented approach works only with process events by grouping events that share common properties, and thereby, grouping together series of crime cases that assists in crime investigations. Further process elements such as control-flow, data, or event logs are not considered. However, in our work we do not consider that information stemming from process events can additionally be included into a dedicated decision model, so this could be done in future works.

The problem of mining decisions in the form of decision rules was addressed in the literature from different perspectives. In [44, 79, 133], a concept of fuzzy rule bases and techniques to design them are introduced. Fuzzy rules learning as introduced in [87, 89, 90] can be applied for solving our problem, but process context and metrics were not taken into account. Fuzzy mining of rules represent a more general case of mining crisp rule bases from data [12, 205].

The comparison of performance of fuzzy against non-fuzzy mining of decision rules can be found in [10] where experiments show that fuzzy classification trees outperform existing decision tree algorithms on 16 real-world datasets. A comparison between a genetic fuzzy and a neurofuzzy classifier results can be found in [88].

Fuzzy mining algorithms are also successfully applied in other domains. Among them they are integrated with genetic algorithms for data classification in database applications in [41]. Also, in [106] they are applied for developing a financial forecasting model, where they are also combined with a genetic algorithm. Minig of fuzzy rules for helping decision making is addressed in [38] for the case of newspaper demand prediction, but the solution is focused on a product, and only a set of fuzzy prediction rules is produced, but not a full explanatory decision model describing the process decision.

With respect to the problem of our thesis, discovering of fuzzy decision models complementary to process models represents a research gap, which we aim at overcoming in Chapter 7. The presented results are based on our work which was published in [24].

3.6 SUMMARY OF RELATED WORK

In this chapter, we examined the work related to our thesis in accordance to its classification structure presented in Figure 16.

In Sections 3.1-3.3, we presented works related to properties of process decisions. Our literature review is based on the decision theory works, from which we derived general decision characteristics. We analyzed which of the derived characteristics are relevant to process decisions which we consider in this thesis, as presented by us in Chapter 2. With respect to the novelty characteristic, process decisions are not unique, as they normally represent repeated operational decisions happening in processes on a daily basis. With regards to the structuredness characterstic, process decisions are normally well-structured and they can be reused in multiple processes. From the point of view of information characteristics, decision models such those expressed with the help of DMN capture decision-related information of any type, but quantitative objectively measurable data is preferable. Representation of elements of decision problem is captured in DMN decision models through both decision requirements and decision logic layers. Decision logic of DMN decision models is most commonly expressed with the help of decision tables, and hit policies of decision table encapsulate output requirements and criteria choice of decision output.

Further, we discussed that decisions are mostly represented in process models through exclusive gateways embedded in control-flow, or by assigning of a business-rule type to activities. We also demonstrated that the paradigm of separation of concerns where decisions are modelled complementary to processes is gaining weight. However, there exist a research gap as there exist no related works addressing concrete implementation of the separation of concerns principle by discovering decision models from process-related information, which we aim to overcome in this thesis.

The literature related to discovery of decision models complementary to process models is presented in Sections 3.4–3.5. Firstly we discussed works on manual design of process decisions. There exist multiple methodologies and guidelines for manual design of decisions, stemming from works on knowledge-based and expert systems, decision support systems, etc. These methodologies could be used for design of decision models, if there exist no process-related information such as process models, data, or event logs. Further, Section 3.5 presents related works on automated and semi-automated discovery of decisions from process-related information. Automating the process of decision model can improve the quality of decision models, their explanatory capability, consistency with the process, and reduced human involvement. We demonstrated that there is also a research gap in the area of extraction of decision models from process models and data. We aim at overcoming this gap by providing corresponding methodologies in Chapters 4 and 5 correspondingly.

Further, we discussed that application of various machine learning techniques can help in automating the process of discovery of decision models from event logs recorded by information systems of enterprises. Although there exist a lot of works on automated derivation of decision rules, none of existing works deal with discovery of decision models which contain not just process decisions, but also underlying decision rules and their dependencies. We propose corresponding techniques to overcome this research gap in Section 6.

Often the decision logic is represented by rules based on Boolean algebra, but the formal nature of is often hard to be interpreted and utilized in practice, because imprecision is intrinsic to real-life decisions. We presented a set of works dealing with incorporation of fuzziness into process and their decision rules. However, we demonstrated that existing works have not considered discovering fuzzy decision models from event logs. In order to overcome this, we propose a methodology for deriving fuzzy DMN decision models from event logs in Chapter 7.

Part II

METHODOLOGIES FOR DECISION MODEL DISCOVERY

4 DISCOVERY OF DECISION MODELS FROM PROCESS CONTROL FLOW

Although many process models from practice contain detailed decision logic encoded through control flow structures, it is not considered good practice. This often results in spaghetti-like and complex process models and reduces maintainability of the models. This chapter presents our methodology to discover decision models complementary to process models from control flow of process models. In particular, we introduce a pattern-based semi-automatic methodology to: (1) identify decision logic in process models, (2) derive a corresponding decision model, and (3) adapt the original process model by replacing the decision logic accordingly. Figure 21 outlines the input and the outputs of the proposed methodology.

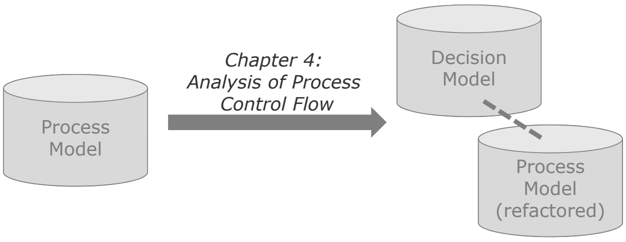

Process Model
Chapter 4: Analysis of Process Control Flow
Decision Model
Process Model (refactored)

Figure 21: Input and outputs of the methodology presented in Chapter 4

The development of the pattern-based methodology to discover decision models complementary to process models is the contribution of this thesis chapter based on results published by Bazhenova et al. in [21] and Batoulis et al. in [18]. The preparation of the real-life data from which the patterns were derived, the derivation of the patterns, and the evaluation of the results is based on a joint work of E. Bazhenova in co-authorship with K. Batoulis, A. Meyer, G. Decker, and M. Weske, published in [18].

The rest of the chapter is structured as follows. Section 4.1 motivates the need to externalize process decision logic into a dedicated decision model. Section 4.2 introduces the methodology to discover decision models from control flow of process model, which is based on the contribution of E. Bazhenova to the results published in [18] and [21]. Section 4.3 introduces a set of decision patterns that can be detected in control flow of process model, which adapts to the formalization of patterns provided for the most part largely by K. Batoulis and A. Meyer in [18]. Section 4.4 is based on the joint work of K. Batoulis, A. Meyer, and E. Bazhenova published in [18] on the identification of the developed patterns in a given a process model. This is followed by the decision model extraction algorithm presented in

Section 4.5, which is based on the contribution of E. Bazhenova in [18] and [21]. Section 4.6 motivates the necessity to adapt output decision and process models and presents the adaptation techniques, which is based on a joint work of all co-authors from [18]. The chapter is concluded by a discussion of the presented methodology.

4.1 INTRODUCTION

In order to get insights whether there exist any patterns of how companies model decision logic in processes, we conducted an analysis of about 1000 real world process models from, amongst others, insurance, banking, and health care, as presented in [18]. All the considered process models were designed with the help of BPMN.

Our analysis results show that part of the logic leading to decisions is often encoded in process models resulting in models that are hard to interpret and maintain. The BPMN standard allows to represent decisions and their impact respectively. However, BPMN is not meant to represent the detailed decision logic since modeling the decision logic often results in spaghetti-like models. For instance, Figure 22 represents a process model of assigning credibility level to a customer. The process is triggered by the start event *Customer credibility request*. Further, the request is processed by conducting the *Initial check* activity. If the customer is a legal body, the constituent documentation is checked. If the documentation is legitimate, credibility level 1 is assigned. Otherwise, credibility level 2 is assigned. If the customer is a private individual, a number of corresponding checks is done. Again, two possible credibility levels can be assigned, as depicted in the figure.

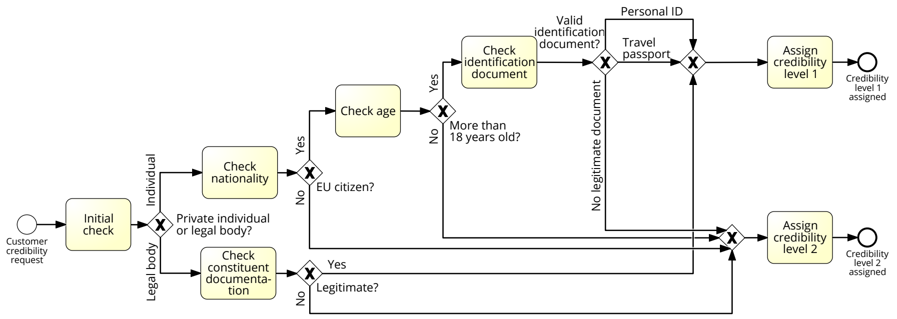

Customer credibility request
Initial check
Individual
Legal body
Private individual or legal body?
Check nationality
Check constituent documentation
Yes
No
EU citizen?
Yes
No
Legitimate?
Check age
Yes
No
More than 18 years old?
Check identification document
Valid identification document?
Personal ID
Travel passport
No legitimate document
Assign credibility level 1
Credibility level 1 assigned
Assign credibility level 2
Credibility level 2 assigned

Figure 22: BPMN model from banking domain illustrating misuse of process control flow for modeling decision logic

It can be seen from the figure, that the decision logic of the process is expressed in control flow through a set of exclusive split gateways. The problem with such encoding of decisions in control flow is that the "nests" of the gateways structures are hard for people to read. Moreover, such process models are not flexible: once they are implemented in information systems, a change in underlying decision logic would require a significant effort in changing the code [197]. Due to these reasons, the OMG consortium recommends to use the separation of concerns principle (see Section 2.3.3) by externalizing the decision logic to a corresponding decision model.

Our analysis of the industrial process models from our partner shows that data-based decisions are most common. A data-based decision is represented by a decision structure consisting of single and compound decision nodes we refer to as split gateways representing exclusive or inclusive alternatives based on external information. These decisions can be classified into three types:

- An explicit decision activity with succeeding branching behavior, e.g., the decision about a customer's age is taken in the decision activity and, based on the result, credibility levels can be assigned.
- Branching behavior is encoded in decision points, e.g., split gateways with decision logic about the customer's age encoded in annotations to the gateways or to edges originating from such gateway.
- There exists a decision activity without succeeding branching behavior, e.g., set discount for a customer based on her age.

As discussed, in Section 2.3, BPMN's scope comprises the business logic containing information on what activities need to be executed in which order by which resource utilizing which data objects. By contrast, DMN covers the decision logic modeling by specifying which decision is taken based on which information, where to find it, and how to process this information to derive the decision result. Since both worlds existed long without proper integration, organizations misused BPMN by including decision logic into process models.

While separation of concerns is easy for newly modeled processes, the existing ones need to be kept usable as well. Otherwise, the migration overhead is too large and organizations retain their BPMN misuse. In the upcoming sections, we introduce means to semi-automatically identify such misused decision logic fragments in a BPMN model and to derive the corresponding DMN model based on information given in the process model. For utilizing the advancements of simplicity and easy maintainability, the original process model gets adapted to replace the decision logic fragment with a reference to the DMN model after its creation.

4.2 METHODOLOGY AND ASSUMPTIONS

Figure 23 visualizes the three main steps of our methodology and the corresponding input and outputs. Given a BPMN model, we first identify decision logic patterns based on the specifications of Section 4.3. Based on our review of a set of real-world process model repositories presented in Section 4.3.1, we defined three decision patterns based on the occurrence of activities and gateways as specified in Section 4.3.2. However, we utilize such information in the refactoring or the post-processing steps which follow in this order upon identification completion.

The identification step (see Section 4.4) is completed if a stakeholder approves the identified patterns to be decision structures. Herewith, multiple patterns may match for parts of the process model such that the stakeholder also must decide which is the appropriate pattern. The decision model extraction is presented in Section 4.5 and comprises the translation of the identified patterns into a DMN model and the adaptation of the process model.

The post-processing step, see Section 4.6, enables configuration of the resulting BPMN and DMN model. Two configuration options and their automatic application are discussed in the corresponding section. Finally, both the BPMN process model and DMN decision model are the outputs of our semi-automatic methodology.

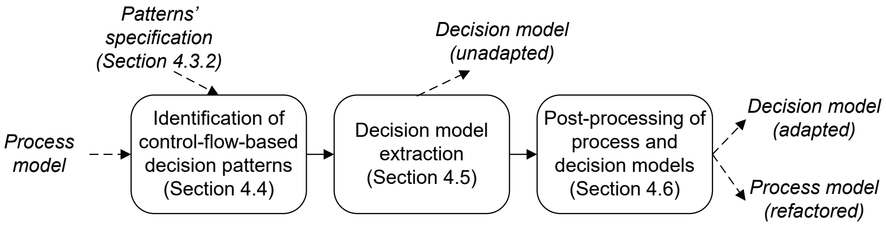

Patterns' specification (Section 4.3.2)
Decision model (unadapted)
Process model
Identification of control-flow-based decision patterns (Section 4.4)
Decision model extraction (Section 4.5)
Post-processing of process and decision models (Section 4.6)
Decision model (adapted)
Process model (refactored)

Figure 23: An outline of our methodology to discover decision models from control flow of process models presented in the rest of Chapter 4

Our analysis of the industrial repositories show that the majority of them contain data-based decisions (cf. 4.3.1). Following the empirical results, in our work, we focus on process models with data-based decisions that are taken within activities directly preceding a split gateway where this gateway only routes the process flow based on the taken decision. Figure 24 presents such decision structure consisting of a single gateway in an insurance environment. Section 4.3.2 introduces three variants of this decision structure which essentially constitutes the control-flow-based decision patterns.

Our methodology to derive decision models from control flow of process models rely on two general assumptions on the process model:

1. A process model is structurally sound, i. e., *pm* contains exactly one start and one end event and every node of *pm* is on a path from the start to the end event.

2. Decisions are taken in distinct process model activities such that a split gateway only routes the process flow based on precalculated decision values, since it is good modeling practice.

Omission of the decision activity preceding the split gateway is a common mistake made by inexperienced BPMN users [197].

When describing identified control-flow-based patterns (see Section 4.3), we refer to activities described in the last assumption as to de facto decision activities, independently of the types assigned to them (cf. Definition 9). Note that these de facto decision activities are not assigned with any type in our illustrations of contol-flow-based decision patterns. Nevertheless, describing the post-processing techniques for process models in Section 4.6, we state that these de facto decision activities should be assigned with a business rule, or a human type, so that the output process model complies to our definition of a decision-aware process model (cf. Definition 14).

4.3 CONTROL-FLOW-BASED DECISION PATTERNS

In order to get insights how decision logic is modeled in real-life environment, in Section 4.3.1 we provide an examination of the real-life process model repositories provided to us by our industry partners, and identify the most common decision-related control-flow structures. Based on this analysis, in Section 4.3.2 we present our specification of control flow patterns which is used further for extraction of decision models complementary to process models.

4.3.1 *Review of the Real-Life Process Model Repositories*

Our review consisted of an analysis of eight industry repositories of process models from the domains of insurance, energy, health care, information technology, banking, and quality control. In total, 1116 process models were reviewed. An overview of our analysis of decision logic representation in the reviewed repositories is presented in Table 5 by industrial segments.

The analysis of the repositories was conducted by us as follows. Firstly, we conducted a manual examination of several process models for finding process model elements involved in modeling the process decision making. Secondly, we conducted an automated counting of process model elements identified in the first step in all the models of the reviewed repositories.

Our initial manual examination of several process models showed that the process decisions in the reviewed repositories were mainly encoded in the control flow of process models. Therefore, we decided to focus only on control-flow-based decision making and excluded the data flow out of the scope of the conducted analysis (for consideration of data-based decision making in process models, see Chapter 5).

Industry	Number of Process Models	Number of gateways					
		XORs		IORs		Event-based	
		AVG	RF	AVG	RF	AVG	RF
Health insurance	163	4,48	96,93	0,09	6,75	0,11	7,36
Energy supplier A	226	5,03	80,09	0,08	3,98	0,16	8,85
Energy supplier B	328	2,54	85,98	0,34	17,38	0,12	9,45
Health care	161	2,81	72,67	0,11	5,59	0,06	5,59
IT governance	14	4,57	100,00	0,21	14,29	0,14	14,29
ISO	68	2,91	54,41	0,03	2,94	0	0
IT service management	153	4,58	71,90	0,01	1,31	0	0,00
Banking	3	3	100,00	0	0,00	0	0,00
Weighted average	–	3,7	80,83	0,15	8,24	0,09	6,63

Table 5: An overview of our analysis of decision logic representation in the real-life repositories obtained from our industrial partner

As discussed in Section 2.3.1, the control-flow structures related to decision making are split gateways. Therefore, at the next step we conducted the automated statistical analysis of the repositories and counted the presence in the process models of three types of split gateways: (1) Exclusive (XORs); (2) Inclusive (IORs); and (3) Event-based. Table 5 shows the average occurrence (AVG) of each gateway type per process model and the relative frequency of occurrence (RF) in percent for each industry sector.

In general, a pattern is a concept that organizes knowledge related to "a problem which occurs over and over again in our environment, and then describes the core solution to that problem, in such a way that you can reuse this solution" [8]. It can be seen from Table 5 that the inclusive (IOR) and the event-based gateways are met in less than 10% of the considered process models which is too low occurrence value for being considered as a reusable decision pattern. Therefore, do not consider such types of gateways for being included into our patterns specification.

At the same time, our analysis shows that that XOR gateways are the most important construct for decision modeling because they occur in more than 80% of all business process models (cf. Table 5). Thereby, we considered decision structures incorporating XOR gateways for being viewed as control-flow-based decision patterns. The manual investigation of process models containing such structures shows that two additional considerations have to be taken into account: (1) the decision should happen in the activity preceding preceding the XOR gateway; (2) decision structures involving XORs can contain multiple outgoing sequence flows; (3) decision structures involving XORs can contain a sequence of split XOR gateways; and (4) a sequence of XOR gateways can be separated by an activity that can belong to the same process decision. We explain these considerations in detail in the specification of control flow patterns related to process decision making presented in the next chapter.

4.3.2 *Patterns' Specification*

Based on our analysis of the real-life process model repositories described in the previous section, we chose the top-three most frequently encountered control-flow-based decision structures. In this section, we define three patterns which are based on these structures.

Each of the presented patterns is represented by a process fragment pf (cf. Definition 7) that can be observed in a process model pm (cf. Definition 8). We assume that each pattern is a process fragment pf that consists of a single start node being an activity, multiple end nodes being activities (one for each alternative), each node is on a path from the start to some end node, and all nodes are not a merge gateway. Also, we assume that XOR and IOR are possible control flow constructs of a process fragment pf referring to an exclusive choice or an inclusive choice respectively. A gateway with multiple incoming and outgoing edges is transformed in two succeeding gateways with one representing the merge and the other the split in this order.

The structure of the description of the patterns is following for all the patterns. Firstly, we introduce the pattern informally on an example process fragment, and then provide a semi-formal definition of it. The detailed formal definitions of the presented patterns are provided in Appendix A. For each pattern, we provide a reference to the corresponding formal definition from this appendix. All the examples represent modifications of the use case of providing a discount for a certain product to the client (see Figure 5).

P1 – Single Split Gateway

Pattern P1 is a process fragment pf that contains a decision structure of an activity preceding a single split gateway with at least two outgoing control flow edges. On each path, an activity directly succeeds the split gateway. Thereby, pattern P1 subsumes optionality decisions as special case; paths directly connecting split and merge gateways get automatically extended by τ-transitions. We assume that the decision is taken in the activity preceding the gateway (cf. Assumption 2 from Section 4.2).

A formal definition of pattern P1 is provided in Appendix A by Definition 28.

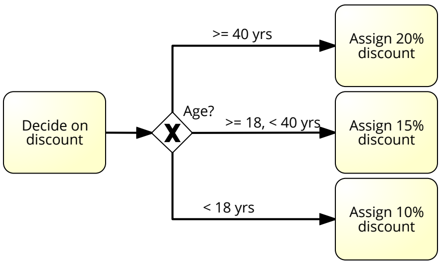


Figure 24: Process fragment representing a split XOR gateway with more than 2 outgoing edges

An example of pattern P1 is given by a process fragment in Figure 24 which consists of a gateway with three alternative paths following it. The gateway is of type XOR, therefore, only one alternative can be chosen. Based on the result of activity *Decide on discount*, i. e.the taken decision about the customer's loyalty, the discount assigned to the customer is set to 20%, 15%, or 10% respectively. Possible results of the decision are fixed by the annotations on the edges originating from the split gateway.

P2 – Sequence of Split Gateways (Decision Tree)

Pattern P2 is a process fragment pf that contains a decision structure of an activity preceding a split gateway with at least two outgoing control flow edges. Thereby, on each path, an activity or another split gateway with at least two outgoing control flow edges directly succeeds the split gateway. In case of a gateway, this proceeds iteratively until all paths reach an activity; i. e.on each path from the first split gateway to some end node of the fragment, there exists exactly one activity – the end node. In accordance to Assumption 2 from Section 4.2, we assume that the decision is taken in the activity preceding the first split gateway.

A formal definition of pattern P2 is given by Definition 29 in Appendix A.

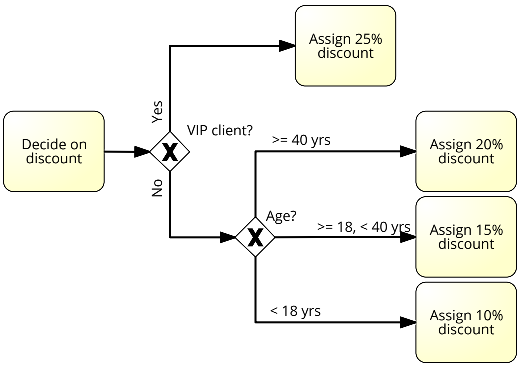


Figure 25: Process fragment representing a sequence of split XOR gateways that represents a decision tree

Figure 25 is an example of pattern P2 which shows a corresponding process fragment with altogether four alternative paths after the first split gateway. Since, all gateways are of type XOR, only one alternative can be chosen. The actual routing based on the taken decisions is distributed over two split gateways. The first routing decision is based on whether the customer is a VIP client. If this is the case, the discount assigned to the customer is set to 25%. Otherwise, the second routing decision is taken based on the age of the customer, and the corresponding discounts are assigned as depicted in the figure. Due to the dependency of a routing decision on the ones taken before, this pattern represents a decision tree. Analogous to pattern P1, the possible results of the decision are fixed by the annotations on the edges originating from some split gateway.

P3 – Sequence of Split Gateways Separated by an Activity

Considering that given is a process model pm, pattern P3 is a process fragment pf that contains a decision structure of an activity preceding a split gateway with at least two outgoing control flow edges. On each path, an activity or another split gateway with at least two outgoing control flow edges directly succeeds the split gateway. An activity that succeeds a split gateway may be succeeded by another split gateway. Otherwise, it is an end node of the process fragment. Activities of type subprocess are also end nodes of the fragment. Iteratively, this proceeds until all paths reach an activity that is not succeeded by some split gateway.

A formal definition of pattern P3 is provided in Appendix A by Definition 30.

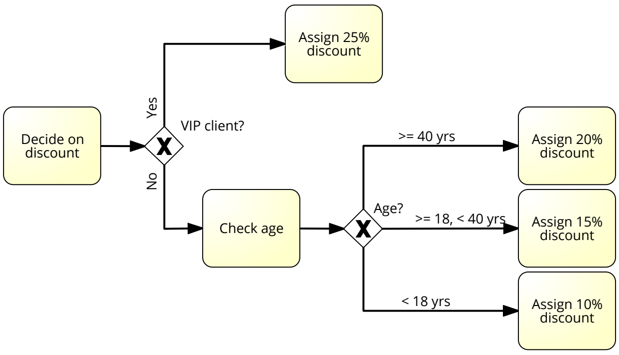


Figure 26: Process fragment representing a sequence of split gateways separated by an activity

Figure 26 presents a corresponding process fragment with altogether four alternative paths after the first split gateway. Since all gateways are of type XOR, only one alternative can be chosen. Each activity of this process fragment that is succeeded by a split gateway (activities *Decide on discount* and *Check age* in Figure 26) takes the actual decisions for the subsequent routing decisions. In case there exist multiple split gateways (see decision tree in pattern P2), the activity takes the decisions for the whole decision tree. This means, this pattern can be composed of multiple decision trees as well as single split gateways. Since multiple decisions are arranged in sequence, we consider this structure as additional pattern to preserve the decision dependencies instead of handling each decision separately. In Figure 26, the choice between a discount of 20%, 15%, and 10% discount is taken based on two decisions (*VIP client* and *Age*) while granting 25% discount is clear after the first decision on whether the client is VIP or not.

4.4 IDENTIFICATION OF THE SPECIFIED PATTERNS IN A PROCESS MODEL

In this section, we describe our approach to identify the control-flow-based decision patterns in a given process model. The overview of the corresponding procedure is depicted in Figure 27.

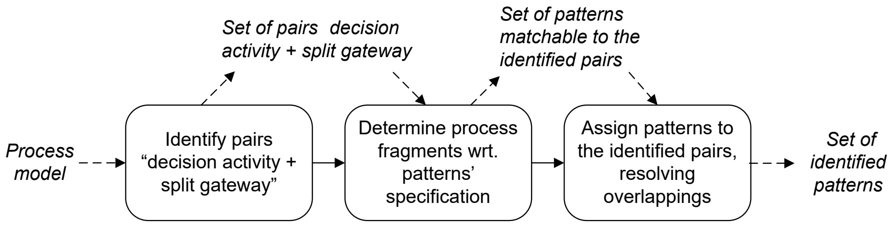

Set of pairs decision activity + split gateway
Set of patterns matchable to the identified pairs
Process model
Identify pairs "decision activity + split gateway"
Determine process fragments wrt. patterns' specification
Assign patterns to the identified pairs, resolving overlappings
Set of identified patterns

Figure 27: Step-by-step identification of the control-flow-based decision patterns in a process model

The first step of the identification process is to determine for all pairs of directly succeeding control flow nodes where the first one is the decision activity and the second one is a split gateway. For each such pair of nodes, we traverse forward the process model and check for existence of a control flow structure aligning to the patterns defined above.

For the detection of pattern P1, we check whether each path originating from the split gateway contains an activity; such fragment is referred to pattern P1. Otherwise, the identification of the pattern is stopped.

For detecting pattern P2, we traverse forward on each path until we identify a non-split-gateway control flow node, e.g., an activity or a merge gateway, directly succeeding a split gateway. The initial split gateway must be followed by at least one other split gateway. Otherwise, the identification of the pattern is stopped.

For detecting pattern P3, we traverse forward on each path until we identify an activity that is directly succeeded by some control flow node that is no split gateway or until we identify a subprocess directly succeeding a split gateway. In case a split gateway is not succeeded by an activity or another split gateway or if an activity is not succeeded by a split gateway, identification of pattern P3 is stopped.

If a fragment is determined, it is referred to the corresponding pattern it was checked for, if it was not stopped. After checking each determined pair, all fragments that refer to some pattern are presented to a stakeholder. Thereby, it is required to decide which actually represent a decision, as the process fragments may overlap.

For example, consider the process fragment pf depicted in Figure 26, which can be referred to pattern P3. However, there also exist two other fragments pf_1, pf_2 that can be referred to pattern P1, if activities *Decide on discount* and *Check age* are viewed as start nodes of fragments pf_1 and pf_2.

The specification of non-overlapping fragments that actually represent a decision structure concludes the first step of our methodology to extract decision models from process models as visualized in Figure 23.

The procedure of identifying a set of control-flow-based decision patterns in a given process model represents, thereby, a semi-automated step of the methodology presented in this chapter (cf. Section 4.2). The involvement of an expert is needed to resolve the possible overlapping of the patterns that can be discovered in an automated way.

4.5 DECISION MODEL EXTRACTION

Next, we discuss the derivation of decision models from a process fragment satisfying one of the patterns presented in Section 4.3. In Section 4.5.1, we introduce a mapping of BPMN constructs which can be met in these patterns, to DMN contructs. Section 4.5.2 presents an exemplary extraction of a decision model from a process fragment.

4.5.1 *Mapping of BPMN and DMN constructs*

The decision encoded in the process fragment representing one of the control-flow-based decision patterns from Section 4.3 is partitioned into a top-level decision connected to sub-decisions with optional input data in DMN. If the decision logic is visible in the process model, we also provide associated decision tables. For this purpose, we devised the rules presented in Table 6 of corresponding model elements that dictates how both the decision requirements level and the decision logic level are constructed.

Below we introduce a *correspondence relation* $M = \{M_1, \ldots, M_6\}$ which relates the BPMN constructs that compose the control-flow-based decision patterns presented in Section 4.3, to DMN constructs. The correspondence relation is visualized with the help of *correspondence graphs* in Table 6. The BPMN constructs represent process fragments (see Definition 8). Hereby, $da \in DA$, $DA \subseteq T$ represents a decision activity from the given process model pm. The DMN constructs represent both DRD fragments and decision tables. All the DRD fragments are subgraphs of a DRD (cf. Definition 12), such that $d{\in}D, ID'{\subseteq}ID$. The decision table definition is given in Definition 13. Thus, the correspondence relation M is defined by us as follows.

M1 A mapping *M1* is a correspondence relation which shows that data-based split gateway g is mapped to DMN decision element d because often the data on which the routing is based results from a decision. For example, in Figure 26, the value of the *VIP client* gateway may need to be inferred from other data such as the number of purchases made so far. Contrarily, note that the value of the *Age* gateway in Figure 26 can be observed directly so that in this case the gateway does not need to be mapped to any DMN element. However, since it is hard to differentiate

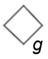
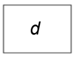
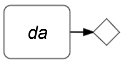
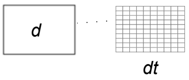
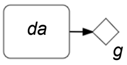
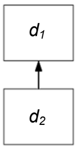
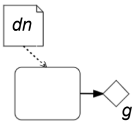
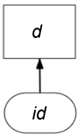
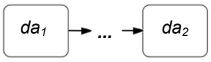
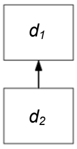
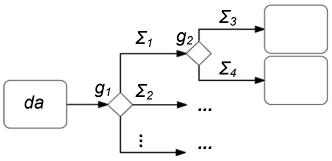
<table>
<tr><th>BPMN construct</th><th>Correspondence graphs</th><th>DRD construct</th></tr>
<tr><td>
</td><td>M1
$g \rightarrow d$</td><td>
</td></tr>
<tr><td>
</td><td>M2
$da \rightarrow d$
$da \rightarrow dt$</td><td>
</td></tr>
<tr><td>
</td><td>M3
$da \rightarrow d_1$
$g \rightarrow d_2$</td><td>
</td></tr>
<tr><td>
</td><td>M4
$g \rightarrow d$
$dn \rightarrow id$</td><td>
</td></tr>
<tr><td>
</td><td>M5
$da_1 \rightarrow d_1$
$da_2 \rightarrow d_2$</td><td>
</td></tr>
<tr><td>
</td><td>M6
$da \rightarrow dt$
$G=(g_1,g_2) \rightarrow I=(I_1,I_2)$
$\Sigma=(\Sigma_1,\Sigma_2,\ldots) \rightarrow Q=(q_1,q_2,\ldots)$</td><td>dt
<table>
<tr><th>I_1</th><th>I_2</th><th>O</th></tr>
<tr><td>q_1</td><td>q_3</td><td>O_1</td></tr>
<tr><td>q_1</td><td>q_4</td><td>O_2</td></tr>
<tr><td>q_2</td><td>⋮</td><td>⋮</td></tr>
<tr><td>⋮</td><td></td><td></td></tr>
</table></td></tr>
</table>

Table 6: Mapping of BPMN constructs which can be encountered in control flow patterns presented in Section 4.3, to DMN constructs. The labeled constructs are affected by some mapping while the non-labeled constructs set the context where required.

these two situations automatically, the default is to map gateways to independent decision elements. The stakeholders can then decide during post-processing whether or not this is necessary, as described in Section 4.6.

M2 A mapping *M2* is a correspondence relation which shows that BPMN decision activity *da* (preceding a gateway) is mapped to DMN decision *d*, which is additionally associated with decision table *dt*. Notice that we are able to specify decision tables for decision activities (mapping *M2*) but not for gateways (mapping *M3*). This is because the concrete value of the variable on which the gateway routing is based usually is set by the activity preceding it, and we cannot derive how this is done by only looking at the process model. In case of decision activities, the situation is different since we can follow each path starting from activity da and ending at another activity and thereby construct a decision table, as will be explained in mapping *M3*.

M3 Because the decision table associated with a decision activity will contain the value of the gateway variable, we map the connection of decision activity da and a succeeding gateway g in BPMN to a decision dependency between d_1 and d_2 in DMN. This is shown in mapping *M3* representing another correspondence relation between BPMN and DMN constructs.

M4 A mapping *M4* is a correspondence relation which illustrates how BPMN data node dn is represented in DMN, if it is available. As just mentioned, we assume that the decision activity sets the following gateway's variable. If a data node is connected to this decision activity, we assume that the data node is used to arrive at this value. Consequently, in the decision model, the data node is mapped to a DMN input data element providing input to the decision element corresponding to the gateway.

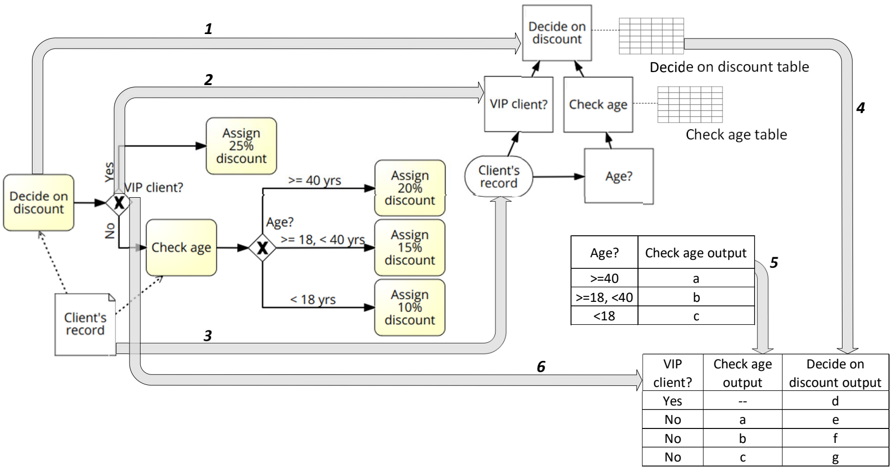
M5 A mapping *M5* is a correspondence relation similar to mapping *M4*. Decision activity da_2 succeeds da_1 without further decision activities in-between them. Then, in the decision model, decision d_1 uses the output of d_2 as input.

M6 A mapping *M6* is a correspondence relation which indicates how decision table dt is derived from the corresponding BPMN construct. The decision table belonging to the da decision activity is composed of all gateways that follow the decision activity. The gateway labels are mapped to column headers, and the edge annotations to corresponding column values. There will be as many rows as there are individual paths starting from activity da and ending at another activity. If any of the gateways on the paths is of the IOR type, the decision table's hit policy (cf. Definition 13) is set to $collect$, since several paths (or rows) can be chosen. Otherwise, the $unique$ hit policy is set. The header of the conclusion column is derived from the decision activity's label and placeholders are used for its cell values. They are used directly in the refactored process model and can be concretized by the process stakeholders during post-processing.

The presented mapping is applied when a pattern from Section 4.3 is found with the corresponding process fragment pf in the given process model pm. Given a process fragment pf, we first identify the decisions, their dependencies and the input data (mappings $1-5$), which altogether constitute DMN decision requirements diagram. Secondly, according to mapping 6, the corresponding decision tables are created. An example extraction of a DMN decision model from a process fragment is given in the next section.

Thereby, the procedure of mapping the detected control-flow-based decision patterns in a given process model, to a decision model, is a step of our methodology (cf. Section 4.2) that can be fully automated.

4.5.2 *Example Extraction*



Age?	Check age output
>=40	a
>=18, <40	b
<18	c

VIP client?	Check age output	Decide on discount output
Yes	--	d
No	a	e
No	b	f
No	c	g

Figure 28: Exemplary mapping for pattern P3: 1 – from BPMN activity to DMN decision; 2 – from BPMN gateway to DMN decision; 3 – from BPMN data node to DMN input data; 4 – from DMN decision table reference to actual DMN decision table; 5 – from DMN rule conclusions of sub-decision to DMN rule conditions; 6 – from BPMN gateway to DMN rule conditions.

This section gives an example illustrating the decision model extraction step described in the previous section using the process fragment satisfying pattern P3 introduced in Section 4.3. The extraction procedure is explained with the help of a Table 6 using the process and decision models of assigning a client discount.

On the left side of Figure 28, one can see the process fragment, whereas on the right side the decision model is shown. The latter is divided into the decision requirements level (top) and the decision logic level (bottom) consisting of decision tables. Also, we inserted arrows to point out the correspondences of the two models' elements. For the sake of clarity, we omitted arrows when the correspondence was already shown by another arrow. For example, Arrow 1 shows that the process model's decision activity *Decide on discount* corresponds to the decision element *Decide on discount* in the decision model (mapping *M2* from Table 6). Consequently, we did not draw an arrow for the decision activity *Check age*.

Arrow 2 illustrates mapping *M1* from Table 6 by mapping the gateway labeled *VIP client?* to an equivalent decision element. The correspondence between BPMN data nodes and DMN input data elements (mapping *M4* from Table 6) is demonstrated by Arrow 3. Note that the connections between the data node and the activities in the process model result in connections between the gateway decision elements and the input data in the decision model.

Furthermore, mapping *M3* from Table 6 is demonstrated by the fact that the *Decide on discount* decision has the *VIP client?* decision as an input requirement. Similarly, corresponding to mapping *M5*, since the task *Check age* succeeds *Decide on discount*, the DMN decision *Decide on discount* also requires *Check age* as an input.

Arrow 4 shows that decisions are connected to decision tables if the decision logic is visible in the process model and Arrow 5 shows that the output column of the sub-decision is used as input column of the dependent decision. Arrow 6 visualizes that the headers of the condition columns correspond to the labels of the gateways following the decision activity and the cell values equal the edge conditions (cf. mapping *M6* from Table 6).

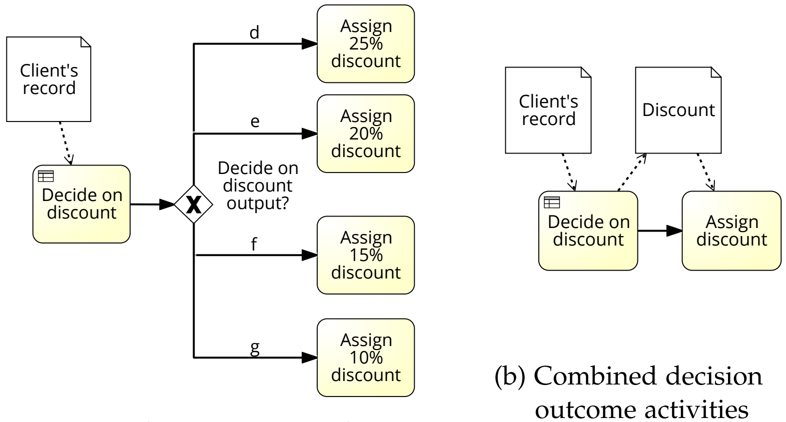
4.6 POST-PROCESSING OF PROCESS AND DECISION MODELS

After extracting the decision logic from a process model to a decision model, both process and decision model need to be post-processed. We devised two ways *I* and *II* to post-process process and decision models, as presented below.

I – Process Fragment Adaptation

According to the specification of decision patterns given in Section 4.3, the entire decision logic should be located in the first decision activity in a pattern. As discussed in Chapter 2, in decision-aware process models, process activities that come to conclusions based on business logic should be assigned with the business rule type (cf. Definition 14). Thus, for the adaptation we transform the activity corresponding to this top-level decision to a business rule type. Since this decision potentially subsumes the decisions corresponding to following decision activities, these activities will not be required anymore in the adapted model. Consequently, we delete each decision activity other than the first from the fragment. This means that also the gateways succeeding the deleted activities can be removed, such that only the first decision activity, the gateway succeeding it and the end nodes of the process fragment are kept. For each end node, the gateway has an outgoing edge connected to it and the conditions with which the edges are annotated equal the row conclusions of the top-level decision table.

An illustration of such refactoring is presented in Figure 29a for pattern P3. It is important to assign the correct conditions to the different edges originating from the split gateway. For example, the end node *Assign* 20% *discount* in Figure 29a is connected to an edge



(a) Partially refactored process fragment

(b) Combined decision outcome activities

Figure 29: Process fragment adaptation for pattern P3

annotated with *e*. This is because in the original process fragment in Figure 28 the conjunction of the conditions leading from the start node to this end node equals $No \wedge (\geq 40\ years)$ and the table row representing this conjunction has *e* as its output value.

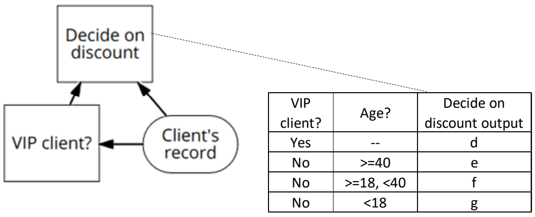
The annotations on the edges originating from a split gateway are intentionally abstract in our approach. The stakeholder may manually add more descriptive annotations by changing the corresponding edge labels or the corresponding rows in the decision activity output column.

After the process fragment is initially refactored, it can be recommended to further simplify process fragments by merging semantically related activities that follow directly after decision-based gateways, with focus on activity.

Considering the adapted process fragment shown in Figure 29a, the activities following on each path represent the same action *Assign discount* based on some data input representing the actual discount value. Choice of this option adapts a process fragment as follows. First, the initial split gateway and all succeeding control flow nodes can be replaced by a single activity whose label the stakeholder has to specify. Secondly, we a data node that is written by the decision activity and read by the newly added activity, can be added. This data node represents the information transferred from the decision to the action taken based on the decision. For the fragment in Figure 29a, the corresponding adapted process fragment is shown in Figure 29b. The added activity is labeled *Assign discount* and the data node is labeled *Discount*.

Investigations on how to automatically merge activity labels for this situation is out of the scope of our thesis. A recommendation for the stakeholders and for future work in this direction would be to consider methods to measure the semantic similarity of the labels which would allow to combine corresponding activities. The works considering automatic retrieval and clustering of similar activity labels include [111, 115, 125].

II - Decision Model Adaptation



VIP client?	Age?	Decide on discount output
Yes	--	d
No	>=40	e
No	>=18, <40	f
No	<18	g

Figure 30: Decision model adaptation for pattern P3: refactored decision requirements diagram and decision table

The decision model, obtained by the decision extraction procedure described in Section 4.5.1, can also be further adapted. An example adaptation is presented in Figure 28. For instance, decisions *Age* and *Check age* can be merged since the first bases on information directly given in the customer information (time being a customer) and requires no computation. In contrast, the *VIP client?* decision requires computation whether a customer is considered loyal and may not be merged with the *Decide on discount* decision. Furthermore, decisions *Decide on discount* and *Check age* could be merged since both contribute to the decision which actual discount shall be awarded to a customer. Figure 30 shows the adapted DMN model based on the discussed decision mergers for the outcome of the P3 fragment given in Figure 29a. Merging decisions requires a merge of the corresponding decision tables, i.e., the dependent decision's table is inserted into the higher level decision's table. The resulting table of merging decisions *Decide on discount* and *Check age* is shown in Figure 30.

In such a way, the post-processing procedure of process and decision models represents a semi-automated step of our methodology from Section 4.2. The refactoring of process fragments corresponding to the detected decision patterns (see Figure 29a for an example) can be fully automated. However, combining the decision outcome activities to avoid the branching behavior of control flow (see Figure 29b for an example) and the decision model adaptation are left for a stakeholder's consideration and execution. After configuring the output models, the stakeholder may adapt the decision tables and the process model a final time.

4.7 SUMMARY AND DISCUSSION

In this chapter, we provided a semi-automatic methodology that allows extraction of decision models from control flow of process models. The extraction is pattern-based derived from an intensive analysis of about 1000 real world process models provided by our project partners. Afterwards, we introduced the procedure of the semi-automated identification of the specified patterns in a process model. Further, we presented the algorithm to derive DMN decision models from BPMN process models.

All the patterns represent process fragments consisting of decision activities and split gateways, since those are the most common decision structures used in real-world process models. Since parallel gateways do not influence decisions and since explicitly considering them would significantly increase the complexity of our methodology and its formalization, we disregard AND gateways.

Although not stated explicit in the patterns, we also support loops like *WHILE x DO y*, since in these cases, the same decision is taken multiple times with varying input data values until the looping condition evaluates to false. Future works on the extraction of decision models from control flow of process models should overcome assumptions that we set for control flow decision structures to identify more patterns and to provide a complete overview about decision logic modeling in process models.

The chapter is concluded by our proposals for the adaptation of the original process model by replacing the decision logic accordingly, and final configuration of the result during post-processing. We use the presented procedures as a base for implementation and evaluation of our methodology which we present further in Chapter 8.

5 DISCOVERY OF DECISION MODELS FROM PROCESS MODEL DATA

As the volume of data is growing, and the methods of analyzing it are constantly evolving, process modeling languages like BPMN demonstrate increasing data awareness. For supporting the separation of concerns during process design, it is crucial to understand which data-related aspects of decision making captured by process models should be externalized to a dedicated decision model. The current chapter extends our methodology of extraction of decision models from control flow of process models presented in the previous chapter, with respect to the data perspective of process models. Under process-related data we understand information that is generated, or consumed by process activities. In particular, we distinguish a set of BPMN patterns capturing process-related data used for making decisions. Then, we provide a formal mapping of the identified BPMN patterns to corresponding DMN models. Figure 31 presents the input and the outputs of the proposed methodology.

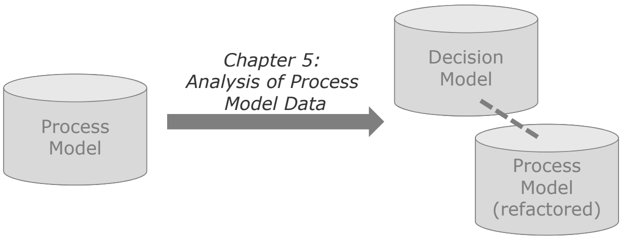


Figure 31: Input and outputs of the methodology presented in Chapter 5

The development of the pattern-based methodology to discover decision models from process model data, presented in this thesis chapter, is based on results published by Bazhenova et al. in [19, 20, 25]. The derivation of the patterns and the validation of the results on an example process from medical domain is a joint work of E. Bazhenova in co-authorship with F. Zerbato and M. Weske, published in [25].

The chapter is structured as follows. Section 5.1 introduces the chapter by presenting a motivational use case demonstrating how process data can be involved in decision making. In Section 5.2, we outline our methodology to discover decision models from process model data, and discuss its assumptions, which is based on the contribution of E. Bazhenova to the results published in [19, 20, 25]. Section 5.3 introduces a review of the BPMN standard from which we derive a set of the data-centric decision patterns, and show how they can be detected in a process model. The aforementioned section is based on a joint work of E. Bazhenova and F. Zerbato published in [25]. Section 5.4 provides a formal mapping between the specifed patterns and a decision model, which is based on the contribution of E. Bazhenova to the results published in [25]. We validate our methodology on the motivating example in 5.5, which is based on the joint work of E. Bazhenova and F. Zerbato, published in [25]. Finally, we conclude the chapter by a discussion of the presented methodology.

5.1 INTRODUCTION

Organizations' value creation is based on information about their own value chain, customers, production, and research and development cycles [202]. Such data is stored in different IT systems that combine the enterprise-wide data utilized in everyday work. At the same time, many decisions in processes are data-driven [129]. As a result, the role of data in process models grew significantly, and it became essential to model the data and its flow within processes. Thereby, process modeling languages that emerged in the last decade, as BPMN, demonstrate increasing data awareness.

In order to provide a mapping between process-related data used for decision-making, and dedicated decision models, different kinds of process-related data used for making decisions needs to be considered. Such data spans from structured operational data to informally defined domain knowledge. As discussed in Section 2.1.2, in this thesis we focus on operational decisions taken in business processes, because they can be well formalized, analyzed, implemented, and reused in multiple processes. Such kind of decisions act on operational data, and, therefore, we consider this type of data for distinguishing different kinds of data-centric process decisions. By operational data we refer to the data that represents the actually manipulated artifacts during the execution of process activities [129].

For providing an insight into how operational data is employed for decision making in processes, let us consider the BPMN process model depicted in Figure 32. This process model presents the initial steps for managing patients with suspected diabetes, at a high level. As shown in Figure 32, the process is triggered by the start message event *Patient Request*, which contains the patient's personal data and the request reason. The request is further evaluated by a *Nurse*, who is the process resource responsible for conducting the *Evaluate Request* decision activity. The nurse evaluates the degree of emergency, according to guidelines extracted from clinical documents and specified in the text annotation attached to the decision activity. If the patient does not require urgent treatment, the nurse proceeds with *Nurse Assessment*, which consists in examining the patient and writing the *Nurse Report*, represented as a data node. This report is later used by a

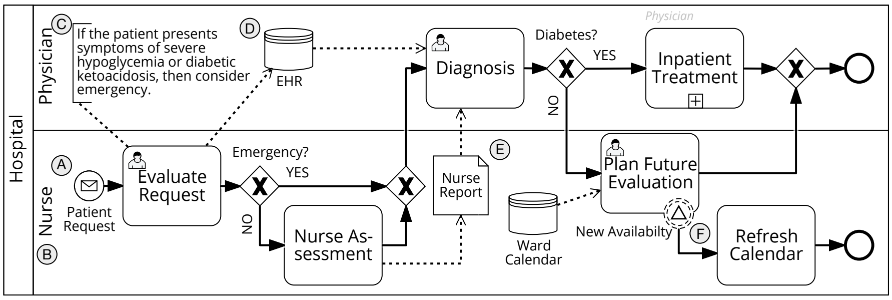


Figure 32: Diagnostic process for patients with suspected diabetes conducted in a healthcare institution. Several data kinds are used by process activities for decision-making: Ⓐ start message event models; Ⓑ resources; Ⓒ text annotations; Ⓓ data stores; Ⓔ data nodes; and Ⓕ boundary non-interrupting event models.

Physician for conducting the *Diagnosis* decision activity, in conjunction with the patient data retrieved from the *Electronic Health Record (EHR)*, shown as a data store. The outcome of the diagnosis determines if the patient requires *Inpatient Treatment* or if the he or she is out of danger and the nurse can *Plan Future Evaluation*. In this latter case, the nurse decides when the evaluation should be scheduled, based on the patient conditions and on the physician's availability, recorded in the *Ward Calendar*. If a *New Availability* notification arrives during the scheduling of the appointment, as depicted by the corresponding boundary non-interrupting event, the time slot is rescheduled.

The process model from Figure 32 considers data carried by BPMN data nodes, text annotations, data stores, events, and resources. For supporting the separation of concerns, the process decision logic need to be externalized to a dedicated decision model. However, it is not immediately clear which process data carries decisional value, and how the various kinds of process-related data should be differentiated when externalized to a decision model. In order to help stakeholders to overcome this problem, we distinguish a set of data-centric patterns representing process decisions and show how to map such patterns to decision models, based on examples of BPMN and DMN.

5.2 METHODOLOGY AND ASSUMPTIONS

Introduction of data-centric decision patterns aims at externalizing the decision logic hidden in process models. The challenging part in the identification of such patterns is that, as also discussed in Chapter 4, we observed that process models from practice often misuse control flow to encode decision logic [18]. In such a way, industrial companies tend to overlook data structures to be involved into process decision making. It is worth noticing that all companies that provided their process models for the study received a respective training from the vendor providing them the BPM modeling tool, and they were aware of the BPMN modeling guidelines [43, 126]. Apparently, the guidelines on modeling data flow in process models *with respect to decision making* are not sufficiently clear.

To overcome this problem, we conduct a systematic qualitative analysis of the BPMN standard in Section 5.3.1 and derive a set of data-centric patterns that can be used for modeling decision making in processes in Section 5.3. We describe how to identify these patterns in a given process model in Section 5.3.3.

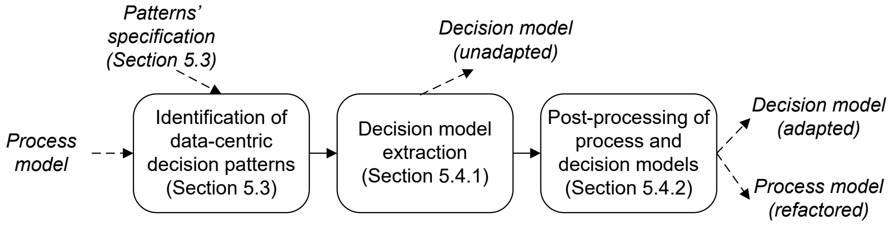

Patterns' specification (Section 5.3)
Decision model (unadapted)
Process model
Identification of data-centric decision patterns (Section 5.3)
Decision model extraction (Section 5.4.1)
Post-processing of process and decision models (Section 5.4.2)
Decision model (adapted)
Process model (refactored)

Figure 33: An outline of our methodology to discover decision models from data of process models presented in the rest of Chapter 5

The process of discovering decision models from decision-related process data from process models is presented in Figure 33. Given a process model, in order to conduct data-centric pattern-based discovery of a decision model from the process model, three steps should be done: (1) Identify the specified decision patterns in a process model; (2) Extract a decision model; (3) Post-process both process and decision models.

To extract data-centric decision patterns from a given process model, we rely on the following assumption. There should exist a mechanism interpreting which activities of a given process model are decision activities. As a first step, we can consider BPMN activity types (cf. [138]) to identify whether an activity is of a business rule, or user type (cf. Definition 9). However, user activities can also represent activities which are not involved in decision making. Furthermore, natural language processing of activity labels [111] can be done with the goal to determine whether an activity is of a decision type. Even in this case, the support of an expert might be needed, as the labels of process model activities are succinct pieces of text, which sometimes do not reflect the decision-making nature of an activity in "as-is" process models. We assume that by any of the proposed ways, the decision activities of a given process models are established.

5.3 DATA-CENTRIC DECISION PATTERNS

In order to identify which combinations represent data-centric decision making in processes, we conducted a systematic qualitative review of the standard as described below. We use BPMN as it is the most consolidated standard for business process modeling [60, 173] and propose a formalization of the patterns. Firstly, we conduct a review of the BPMN standard in Section 5.3.1. From this review, we derived a set of elementary decision patterns representing combinations of decision activities and other process model elements in Section 5.3.2.

5.3.1 *Review of the BPMN Standard*

In order to identify which combinations represent data-centric decision making in processes, we conducted a systematic qualitative review of the BPMN standard [138] and the guidelines for its usage [43, 206]. The review results are presented below.

Our review question is "which BPMN elements represent data in process models that can be related to decision-making activities?". The goal is to provide a selection of elements relevant to the review question. On the one hand, there are data flow elements (e.g., data nodes) which can be connected to decision activities. On the other hand, there are control flow elements (e.g., control flow edges) that can connect decision activities with other types of data represented in a process model (e.g., events providing data for decision activities). The source of the review data is version v.2.0.2 of the BPMN standard [138]. In total, all 116 elements of the standard were considered.

The five basic groups of BPMN elements are: (1) flow objects, (2) data, (3) connecting objects, (4) swimlanes, and (5) artifacts. The definitions of many of the elements can also be found in Section 2.3.1. The graphical illustrations of some of the elements can be seen in Figure 6. A short review summary is presented in Table 7. In the table, the relevance of the BPMN elements for representing data that can be related with decision activities is denoted by "+" symbol for the full relevance, by "-" symbol for the full irrelevance, and by "+/-" symbols for the partial relevance, that is explained in the commentaries column. We consider each basic category with respect to our review question as follows:

Groups	Elements	Relevance	Commentaries
Flow Objects	Event models	+/-	Only start event models, intermediate catching event models, and boundary event models are relevant. Boundary interrupting event models: the decision activity is located on the outgoing exception flow. Boundary non-interrupting event models: attached to a decision activity.
	Activities	-	Not relevant: activities are units of work that are not meant to represent data.
	Gateways	-	Not relevant as gateways are part of control flow which is not meant to represent data.
Data	Data nodes Data inputs Data outputs	+	Relevant, combined with data associations that connect them to decision activities.
	Data stores	+	Relevant, in combination with data associations that connect them to decision activities.
Connecting objects	Sequent flows	+/-	Only sequence flow that is connected to decision activities is relevant.
	Conditional flows	-	Not relevant, as they encompass automatic "routing decisions".
	Message flows	-	Not relevant, as decision activities cannot be of send or receive type.
	Associations	+	Relevant, in combination with text annotations.
	Data associations	+	Relevant, in combination with data objects, data inputs, data outputs, or data stores.
	Exception flows	+/-	Relevant only when it connects a boundary event attached to one activity with a following decision activity.
Swimlanes	Lanes	+	Relevant, as it contains data about a resource responsible for execution of decision activities.
	Pools	+/-	Relevant only if a pool consists of one lane because a decision activity can belong to only one lane.
Artefacts	Groups	-	Not relevant because groups represent only an informal visual mechanism for grouping elements, and they do not carry decisional data.
	Text annotations	+	Relevant, combined with associations that connect them to decision activities.

Table 7: Relevance of the BPMN elements for representing data that can be used by decision activities for making a decision. Full relevance w.r.t. our review question is shown as a "+" symbol, full irrelevance is denoted by a "–" symbol, and partial relevance is marked by a "+/–" symbol.

- *Flow objects* are the main graphical elements to define the behavior of a business process. BPMN defines three flow objects:
  - *Events models.* There are three types of event models, based on when they affect the flow: start, intermediate, or end. We consider only such events that affect execution of decision activities. Therefore, only the start and intermediate event models preceding a decision activity are relevant, since the end events can happen only after its execution. Within the scope of start and intermediate event models, we consider catching event models and exclude throwing event models. Indeed, catching events can react to certain triggers that define the cause for the event. If a decision activity is connected to a catching event model, it means that it can utilize the data carried by the event in decision making. Throwing events can also carry data, but they "throw" it to an element which is different from the decision activity (otherwise, a data node should be used). Hereby, we include into our selection start and intermediate event models of any type (e.g., message, or timer), since the decision logic based on data of any nature can be eventually externalized into dedicated decision models.
  - *Activities.* An activity is a generic term for work that company performs. Naturally, activities are not meant to represent data in processes, therefore they are not relevant for our selection.
  - *Gateways.* Gateways are used to control the divergence and convergence of sequence flow in a process. In such a way, gateways are part of process control flow, so they do not represent data in processes. Even if gateways execute data-based routing decision, the actual decision shall happen in the activity preceding the decision gateway. Omission of such decision is a mistake made by an inexperienced user [60, 197].
- *Data* is represented with the following elements:
  - *Data nodes, data inputs, data outputs.* Data nodes provide information about what activities require to be performed, or what they produce. Data inputs and data output provide the same information for processes. Therefore, data nodes, data inputs, and data outputs, combined are naturally relevant to be included in our selection. As we also see later in the description of the connecting objects, the considered data nodes, data inputs, and data outputs should be used in combination with data associations.
  - *Data stores.* A data store provides a mechanism for activities to retrieve or update stored information that will persist beyond the scope of the process. Naturally, data stores provide information to activities relevant for decision making, so we

include data stores in our selection. Again, data stores should be used in combination with data associations.

- *Connecting objects* represent four ways of connecting the flow objects to each other or other information:
  - *Sequence flows.* A sequence flow can be used to show a partial ordering of activities in a process. As mentioned, activities are not relevant for our selection, therefore, combinations "sequence flow plus decision activity" are also not relevant.
  - *Conditional flows.* A conditional flow is a special kind of sequence flow having a condition expression that is evaluated at runtime to determine whether that flow can be used or not. Despite relying on the evaluation of a data-based condition, they encompass an automatic "routing decision". Since decision-wise their behavior is similar to that of exclusive gateways, they are not relevant for our selection.
  - *Message flows.* A message flow is used to show the flow of information messages between two participants that are prepared to send and receive them. Thereby, the send or receive types should perform such information exchange. However, the decision activities shall serve a purpose of pure decision making, and we assume that they can be only of either a business rule, or user (cf. Definition 9). Therefore, we exclude message flows from our selection.
  - *Associations.* An association is used to link text annotation with BPMN graphical elements. Thereby, text annotations can contain data related to process decision making. In such a way, we include combinations of "association plus text annotation" into our selection.
  - *Data associations.* A data association is used to link data nodes or data stores with BPMN graphical elements. Thereby, data nodes or data stores can contain data related to process decision making. In such a way, we include combinations of "data node, or data store plus data annotation" into our selection.
  - *Exception flow.* Exception flow occurs outside the normal flow of the process and is based upon an intermediate event model attached to the boundary of an activity that occurs during the performance of the process. Hereby, we consider both interrupting, and non-interrupting event models, as events of both types can carry data which can be used by a decision activity.
- *Swimlanes* group the primary modeling elements by two ways:
  - *Lanes.* Lanes are used to organize and categorize activities. The assignment of a resource to an activity is done by placing an activity in the lane.

  - *Pools.* A pool is the graphical representation of participants of the process. It can consist of several lanes. A decision activity can belong to only one lane of the pool. Again, the assignment of a resource to an activity is done by placing an activity within the lane. In such a way, the case of pools can be reduced to the case of lanes.

- *Artifacts* are used to provide additional information about the process:
  - *Groups.* BPMN defines groups as a "visual mechanism to group elements of a diagram informally". Groups can not be connected to any BPMN elements, and they do not affect the process flow. In principle, they serve to ease perception of process models to users. Since our goal is to provide a formal specification for data-centric patterns that are relevant for operational decision making in processes, we do not include groups into our selection.
  - *Text annotations.* Text annotations are a mechanism for a modeler to provide additional information in natural language to help the readers of a process model. These artifacts can contain significant information guiding decision making in processes, therefore, we include them into our selection.

Based on the presented review of relevance of the BPMN elements for representation of data that can be related with decision activities, we define next a set of patterns that can be used for further externalization of process decision logic into a dedicated decision model.

5.3.2 *Data-Centric Decision Patterns*

In order to enable the extraction of process decision logic into a dedicated decision model, we define a set of data-centric decision patterns *Π1*–*Π6* that can occur in a process model. The patterns are derived from our systematic review of the BPMN standard (cf. Table 7). Next, we discuss the detection of these patterns in a given process model.

The overview of the discovered patterns is presented in Figure 34. Each pattern corresponds to a process fragment, that is, a subgraph of a process model, which represents a data-centric decision that can be externalized into a separate decision model. For better readability, we always show one process elements of a kind, thus omitting the representation of multiple elements of the same kind connected to a single decision activity. For example, pattern *Π1* shows only one data node connected with a decision activity, although the formalization of *Π1* provides that multiple data nodes can be attached to a single decision activity. Note, that in our illustrations, the decision activities can be of two types: business rule, or user.

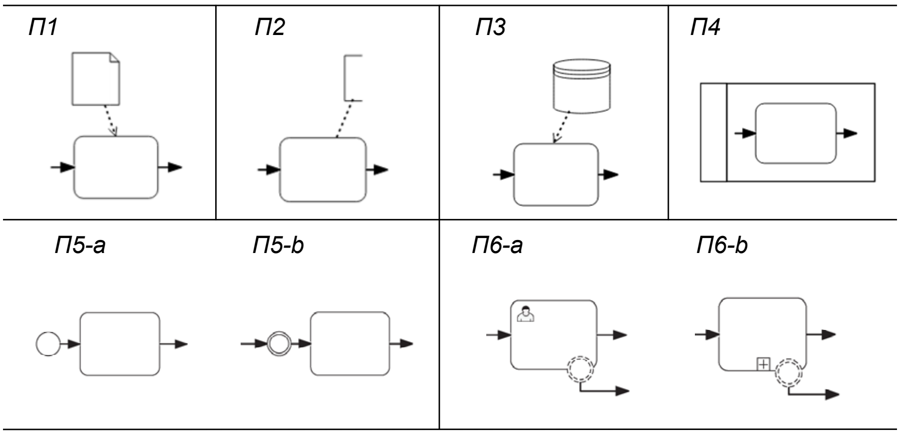


Figure 34: BPMN data-centric decision patterns showing decision activities identified by experts (cf. Definition 2). The possible plurality of the elements connected to a single decision activity (data artifacts, annotations, or boundary event models) is omitted for better readability reasons.

Let pm = $(N, DE, TA, \Sigma, C, F, Z, H, \alpha_s, \alpha_t, \alpha_g, \beta, \xi, \psi)$ be a process model (cf. Definition 7), and let $DA \subseteq T$ be the set of decision activities for pm (cf. Definition 9). A process fragment $pf = (N', DE', TA', \Sigma', C', F', Z', H', \eta_s, \eta_t, \eta_g, \kappa, \sigma, \zeta)$ is a connected subgraph of the given process model pm (cf. Definition 8). We refer to patterns $\Pi 1 - \Pi 6$ as to elementary patterns. An *elementary pattern* is a process fragment which consists of a decision activity and related elements. We devise the formalization of the discovered patterns, as presented below.

Π1 *Data nodes used by a decision activity.* A decision activity $da \in DA$ uses the set of data nodes $DN' \subseteq DN$ iff for each $dn \in DN'$, $(dn, da) \in F$. $\Pi 1$ is a process fragment that consists of decision activity da, a set of data nodes $DN' \subseteq DN$, and a set of data associations $F_{DN'} = \{(dn, da) \mid dn \in DN'\} \subseteq F$.

Π2 *Text annotations used by a decision activity.* A decision activity $da \in DA$ uses the set of text annotations $TA' \subseteq TA$ iff for each $ta \in TA'$, $(ta, da) \in Z$. $\Pi 2$ is a process fragment that consists of decision activity da, a set of text annotations $TA' \subseteq TA$, and a set of undirected associations $Z' = \{(ta, da) | \, ta \in TA'\}$.

Π3 *Data stores used by a decision activity.* A decision activity $da \in DA$ uses the set of data stores $DS' \subseteq DS$ iff for each $ds \in DS'$, $(ds, da) \in F$. $\Pi 3$ is a process fragment that consists of decision activity da, a set of data stores $DS' \subseteq DS$, and a set of data associations $F_{DS'} = \{(ds, da) \mid ds \in DS'\} \subseteq F$.

Π4 *Decision activity associated with a specific resource.* A decision activity da is based on data related to the associated process resource $h \in H$ iff $(da, h) \in \psi$. *Π4* is a process fragment that consists of decision activity da and associated resource h.

Π5 *Event data used by a subsequent decision activity.* A decision activity $da \in DA$ uses the data carried by a previously occurred event $e \in E$ iff $(e, da) \in C'$, where $C' \subseteq C$. According to the kind of triggered event, we distinguish two variants of this pattern:
$\Pi 5 - a$ is a process fragment that consists of decision activity da, start event e, and control flow $(e, da) \in C'$.
$\Pi 5 - b$ is a process fragment that consists of decision activity da, intermediate event e, and control flow $(e, da) \in C'$.

Π6 *Boundary event data used by a decision activity.* A decision activity da may be influenced by the occurrence of the set of boundary events $E'_B \subseteq E_B$ during its execution iff for each $e_b \in E'_B$, $(e_b, da) \in \kappa$, $\kappa : T \rightarrow 2^{E'_B}$, and e_b occurs while da is being executed. This holds only for human decision-making, or for subprocesses, which usually take a certain amount of time to be executed, during which an event can occur and provide data that influence decision outcomes. Indeed, a standalone business-rule decision activity is assumed to invoke a decision model which is executed instantly [142]. Accordingly, we distinguish two pattern variants:

$\Pi 6 - a$ is a process fragment that consists of decision activity da, the boundary event e_b such that $(e_b, da) \in \kappa$, where $\alpha_s(da) = \{task\}$ and $\alpha_t = \{user\}$.

$\Pi 6 - b$ is a process fragment that consists of decision activity da, boundary event e_b such that $(e_b, da) \in \kappa$, where $\alpha_s(da) = \{subprocess\}$.

It is worth noticing that not all kinds of data specified within process models need to be externalized in dedicated decision models. For instance, let us consider event-based gateways. In this case, the event occurrence drives instantaneous decisions, which are managed by process engines. Such kind of *process-engine decisions* should not be included in decision models, as they are rather based on process logic and routing rules, which are out of the control of a (human) decision maker. On the contrary, information used for decision-making and included in decision models may not be explicitly represented in process models. For instance, domain knowledge, Key Performance Indicators (KPIs), or process execution logs often drive decision making, but they are represented as meta-information rather than being included in process models.

5.3.3 *Identification of the Specified Patterns in a Process Model*

The detection of the data-centric patterns in a given process model should be done in the following way. Firstly, according to Assumption 2 given in Section 5.2, a process or decision analyst detects all decision activities. In particular, the analyst identifies the decision activities based on his or her process knowledge, or consults the process participants. A semi-automated support of this step can be done by using natural language analysis of activity labels [111] to recommend candidate decision activities which shall be confirmed by an analyst. Another option for semi-automated support of this step is to identify the split gateways in a given process model, and recommend the activities preceding the gateways as candidate decision activities. Again, an analyst should confirm that the selected activities actually encompass decision making. In practice, there are frequent situations where the decision activities are omitted before split gateways [18].

Next, for each decision activity it is checked whether it is in the control-flow or data relation with other elements in accordance to $\Pi 1 - \Pi 6$. This step can be automated by a corresponding parsing of the process model. The output of the procedure is the set of detected patterns for each decision activity identified by an analyst. Note that each decision activity can be involved in several patterns at the same time. The examples of elementary patterns extracted for the motivating example (see Figure 32) and the detail regarding their detection can be found in Section 5.5.

5.4 DECISION MODEL EXTRACTION

In order to externalize the decision-related process data to a dedicated decision model, a mapping between process and decision model needs to be provided. Section 5.4 introduces the formal mapping between the set of data-centric decision patterns introduced in Section 5.3 and corresponding decision models. In Section 5.4.2, we show how to adapt both process and decision models after the extraction of the decision model.

5.4.1 *Formal Mapping*

As a basis for decision models we use the DMN standard, but we also provide the formalization of the mapping. DMN decision models consist of two layers - the decision requirements diagram (cf. Definition 12) and the decision logic most often represented through decision tables (cf. Definition 13). For extracting a decision model from process model data presented in this chapter, we only utilized the decision requirements level. The reason for this is that by identification of the data-centric decision patterns introduced in Section 5.3, decision activities are identified by analysts independently of their further influence on the process, so that the decision outcomes are not considered. This is the opposite approach to the identification of the control-flow-based decision patterns presented in Section 4.3, because there the decision patterns included decision activities and their output expressed in the control flow. However, the data-centric decision patterns introduced in Section 5.3 consider only the decision activities and their inputs. Thereby, the decision logic is not represented in the combinations of decision activities and process data, so decision tables can not be constructed based on this knowledge. Herewith, the process data can be utilized when detecting the requirements for the decision model. In the remainder of the section, we propose a mapping of the presented data-centric decision patterns to the decision requirements level of the decision model.

Below, we introduce a set of DRD fragments $\Delta = \{\Delta 1, \ldots, \Delta 6\}$ which corresponds to the set of identified BPMN data-centric decision process patterns $\Pi = \{\Pi 1, \ldots, \Pi 6\}$. Further, we provide a *correspondence relation* $\Gamma = \{\Gamma 1, \ldots, \Gamma 6\}$, such that $\Gamma \subset \Pi \times \Delta$. All the DRD fragments are subgraphs of a DRD (c.f. Definition 12) containing one decision d, such that $d \in D, ID' \subseteq ID, KS' \subseteq KS, IR' \subseteq IR, AR' \subseteq AR$. The correspondence relation Γ is visualized with the help of *correspondence graphs* in Figure 35, and discussed below. In Figure 35, the shading of the shapes denotes that the corresponding DRD elements are optional for both decision representation and execution. Again, for readability reasons, we do not show the possible plurality of elements of the same kind connected to a decision activity. In detail, the correspondence relation Γi maps decision activity da of each BPMN decision pattern Πi to decision d of the corresponding DRD pattern Δi, where $i = \{1, ..., 6\}$. Bearing this in mind, we define and discuss the remaining BPMN and DRD elements.

- **$\Gamma 1$** A mapping $\Gamma 1$ is a correspondence relation between the BPMN pattern $\Pi 1 = (da, DN', F_{DN'})$ and the DRD fragment $\Delta 1$=(d, ID', IR'). Thereby, each data node $dn \in DN'$ corresponds to input data $id \in ID'$ as they both represent operational data used by the decision. Each corresponding data association $f_{dn} \in F_{DN'}$ corresponds to information requirement $ir \in IR'$.

- **$\Gamma 2$** A mapping $\Gamma 2$ is a correspondence relation between the BPMN pattern $\Pi 2 = (da, TA', Z')$ and the DRD fragment $\Delta 1 = (d, ID', KS', IR', AR')$. Text annotation $ta \in TA'$ corresponds to input data $id \in ID'$ if it represents operational data needed for making the decision. In this case, undirected association $z \in Z'$ should be mapped to information requirement $ir \in IR'$. Alternatively, text annotation $ta \in TA'$ may be mapped to knowledge source $ks \in KS'$ if it represents a non-functional requirement

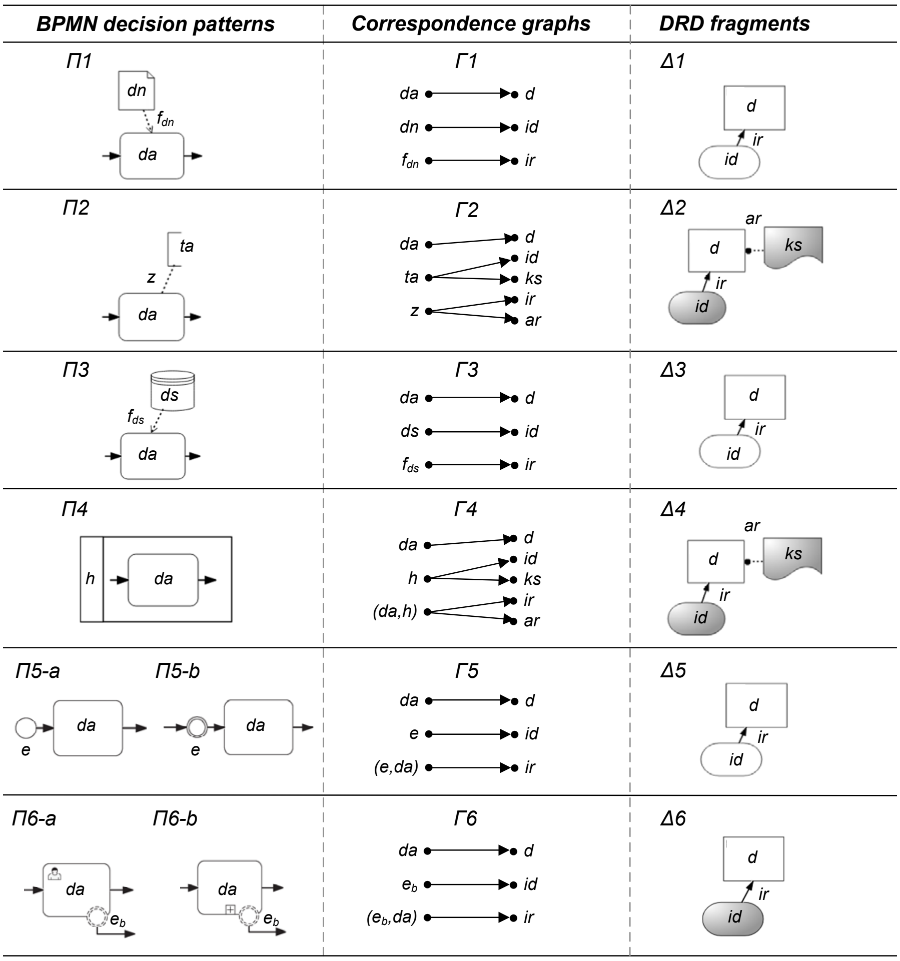


Figure 35: Mapping of the introduced BPMN patterns to DRD fragments. Shading of the DRD shapes means that the elements are optional for modeling and execution.

for decision-making. In this latter case, undirected association $z \in Z'$ should be mapped to authority requirement $ar \in AR'$. However, as text annotations do not always represent data used for making decisions, both input data and knowledge source in the DRD fragment are represented as optional for modeling and execution.

Γ3 A mapping $\Gamma 3$ is a correspondence relation between the BPMN pattern $\Pi 3 = (da, DS', F_{DS'})$ and the DRD fragment $\Delta 3 = (d, ID', IR')$. Each data store $ds \in DS'$ corresponds to input data $id \in ID'$ as it represents operational data used by the decision. Each data association $f_{ds} \in F_{DS'}$ corresponds to information requirement $ir \in IR'$.

Γ4 A mapping $\Gamma 4$ is a correspondence relation between the BPMN pattern $\Pi 4 = (da, h, (da, h))$ and the DRD fragment $\Delta 4 = (d, id, ks, ir, ar)$. Resource h can be mapped to input data $id \in$

ID' if it represents role information used for decision-making. In this case, $(da, h) \in \psi$ should be mapped to information requirement ir. Alternatively, resource h can be mapped to knowledge source $ks \in KS'$ if it represents a non-functional requirement for making the decision, and then $(da, h) \in \psi$ should be mapped to authority requirement $ar \in AR'$.

$\Gamma 5$ A mapping $\Gamma 5$ is a correspondence relation between the BPMN pattern $\Pi 5 = (da, e, (e, da))$ and the DRD fragment $\Delta 5 = (d, id, ir)$. Both $\Pi 5 - a$ and $\Pi 5 - b$ are implied, as they have the same formal structure. Event e carries process data influencing the decision, and it corresponds to input data $id \in ID'$. The corresponding control flow edge $(e, da) \in C'$ is mapped to information requirement $ir \in IR'$.

$\Gamma 6$ A mapping $\Gamma 6$ is a correspondence relation between the BPMN pattern $\Pi 6 = (da, E'_B, \kappa)$ and the DRD fragment $\Delta 6 = (d, ID', IR')$. Each boundary event $e_b \in E'_B$ carries data influencing the decision, and it corresponds to input data $id \in ID'$. As well, the corresponding relation $(e_b, da) \in \kappa$ is mapped to information requirement $ir \in IR'$. As the boundary event in both cases might not occur at all, the corresponding input data element in the DRD fragment is shown as optional.

The decision nodes in the DRD fragments presented above can additionally reference the business knowledge nodes reflecting decision logic. This is recommended if the decision logic is reused by multiple decisions. In this case, the corresponding knowledge requirements nodes should be provided. Also, text commentaries can be added in the output DRD fragments to additionally specify the DRD elements.

Thereby, the procedure of mapping the detected decision patterns in a decision-related data of a process model, to a decision model, is a step of our methodology (cf. Section 5.2) that can be fully automated.

5.4.2 *Post-Processing of Process and Decision Models*

Given that a set of the DRD fragments is consequently derived from the set of BPMN data-centric decision patterns according to the presented mapping, two post-processing steps need to be done. Firstly, a complete DRD model needs to be constructed. This should be followed by a compilation of the derived DRD fragments into a comprehensive decision requirements diagram and by the reduction of repeated elements. When a decision activity produces data nodes or text annotations that are reused by another decision activity, an information requirement should be added between these two decisions. Otherwise, DRD fragment represents an independent DRD model.

The example of construction of a DRD model from derived data-centric process fragments for the motivation example is provided in Section 5.5.

After a DRD model is constructed for a given decision model, the original process model needs to be adapted. Firstly, an analyst should identify whether the process related data elements involved in the specified patterns should be kept in the given process model, or they should only be kept in a dedicated decision model. Secondly, for the determined decision activities of the process model, undirected association links to the corresponding elements of the extracted DRD model should be added, e.g., at the implementation level.

In contrast to the adaptation of a process model following the control-flow-based decision model extraction (as presented in Chapter 4), the process model post-processing in case of the data-centric decision model extraction is not a necessary step. This is due to the fact that we consider that the process model control flow is misused if the decision logic is embedded in the control-flow structures (cf. Section 4.1). However, the decision-related process data is rather represented through process model data elements aiding the decision activities, but not encoding the decision logic.

5.5 APPLICATION OF THE METHODOLOGY TO A REAL-WORLD EXAMPLE

In this section, we apply the methodology introduced for discovering decision models from process model data to a real-world example, introduced in Section 5.1, to show its applicability and validate it.

Pattern Detection

Firstly, we analyzed the BPMN process model from Figure 32 and detected that there are three decision activities which reflect the process of choosing an output from some alternatives, namely *Evaluate Request*, *Nurse Assessment*, and *Plan Future Evaluation*. According to the presented classification of the BPMN data-centric decision patterns from Section 5.3.2, an aggregate of the process fragments corresponding to these decision activities was detected (see Figure 5.5): (I) The *Evaluate Request* decision activity and the start message event *Patient Request*; (II) The *Evaluate Request* decision activity and the text annotation connected to it; (III) The *Evaluate Request* decision activity and the *Nurse* resource; (IV) The *Diagnosis* decision activity and the *Nurse Report* data node; (V) The *Diagnosis* decision activity and the *EHR* data store; (VI) The *Diagnosis* decision activity and the process resource *Physician*; (VII) The *Plan Future Evaluation* decision activity and the *Ward Calendar* data node; (VIII) The *Plan Future Evaluation* decision

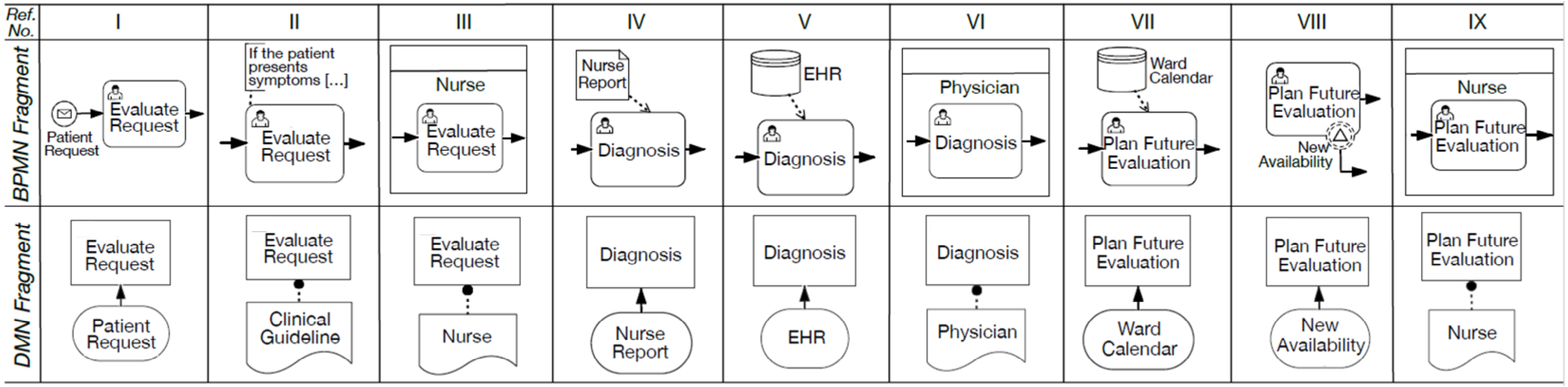


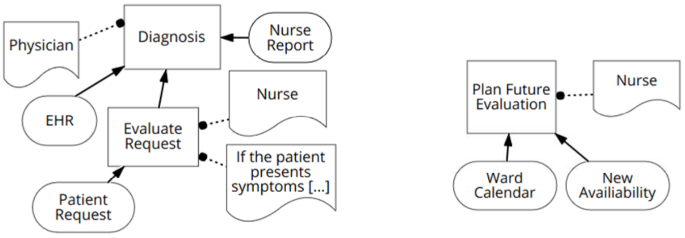
Figure 36: Detected data-centric decision patterns for the BPMN process model from Figure 32 and corresponding DRD fragments

activity and the *New Availability* notification; and (IX) The *Plan Future Evaluation* decision activity and the *Physician* resource.

Further, we applied the mapping from Section 5.4.1 on the detected fragments and provided the corresponding DRD fragments, as shown in Figure 5.5. The cases including optional mappings were treated as follows. In Fragment II, the text annotation is mapped to the knowledge source and not to input data, as it is a non-functional requirement for executing decision activity *Evaluate Request* decision activity. All the resources were mapped to knowledge sources (Fragments III,VI, and IX), as they are constant for the exemplified process model, whereas input data rather reflects changeable operational data. Accordingly, the *New Availability* boundary event model (Fragment VIII) is mapped to the input data.

Post-Processing

The discovered fragments were consequently combined together and compiled into two decision requirements diagrams, presented in Figure 37. As the output of the *Evaluate Request* decision activity is written in the *EHR* data store which is read by the *Diagnosis* decision activity, an information requirement between the corresponding two decisions was added, as shown in Figure 37a. Since the *Plan Future Evaluation* decision activity is connected with the other decision activities only through the process control flow, it is designed as an independent DRD in Figure 37b.



(a) The first extracted DRD

(b) The second extracted DRD

Figure 37: Extracted DMN model corresponding to the BPMN process model from Figure 32

The extracted DMN model in Figure 37 serves as an explanatory decision model for the BPMN process model from Figure 32, as it incorporates explicitly the decision-related process data. Herewith, the extracted decision model can be executed complementary to the process model, and thus, the principle of separation of concerns [197] is supported. Thereby, the original BPMN process model should contain the undirected association links of decision activities to the corresponding elements of the extracted DMN decision model, which can be realized during implementation.

5.6 SUMMARY AND DISCUSSION

In this chapter, we presented a pattern-based methodology to externalize process-related data used for making decisions into dedicated decision models. In general, companies tend to overlook the inclusion of data into process decision making, perhaps, because the guidelines on modeling data flow in process models with respect to decision making are not sufficiently clear. To overcome this gap, we conducted a qualitative analysis of the BPMN standard in Section 5.3.1. This allowed us to derived and formalize the set of data-centric decision patterns that can be used for modeling decisions in process models in Section 5.3. Section 5.3.3 describes how to detect the specified patterns in a given process model.

Similarly to our methodology from Chapter 4 (cf. Section 4.7), the methodology presented in this chapter can handle loops if they are present in an input process model. In such cases, the same decision is taken multiple times with varying input data values until the looping condition evaluates to false.

Our assumption on the process model is that there exists a mechanism interpreting which activities of the given process model are decision activities. In future work, such assumption can be overcome, for example, by improving identification of decision activities in process models based on activity labels and textual content processing [111].

The introduced specification of the data-centric decision patterns allowed us to introduce the pattern-based mapping between the detected set of process model patterns and corresponding fragments of the decision requirements diagrams. The mapping considers the different nature of data artifacts specified in process models.

We demonstrated and validated the presented methodology to discover decision models from process model data by applying it to an example taken from the healthcare domain. The evaluation of the presented techniques is provided further in Chapter 8. In particular, we present results of the application of our methodology to discover decision models from a real-life process model set, and further explore the advantages and limitations of the presented methodology.

6 DISCOVERY OF DECISION MODELS FROM EVENT LOGS

So far, we have addressed the ways of discovering decision models complementary to process models by the analysis of process model control flow and data. However, knowledge about ”as-is” decision making can also be retrieved by analyzing process event logs and discovering decision models from this information. In this chapter, we propose an approach for semi-automatic derivation of decision models from process event logs with the help of decision tree classification. Thereby, we extended a state-of-the-art approach to derive control flow decisions from event logs with additional identification of data decisions and dependencies between them. Figure 38 outlines the input and the output of the proposed methodology.

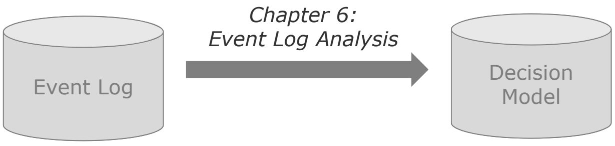


Figure 38: Input and output of the methodology presented in Chapter 6

The development of the methodology to discover decision models complementary to process models from event logs is the contribution of this thesis chapter based on results published by Bazhenova et al. in [21] and [23]. The development of the algorithms from this chapter and their implementation is a joint work of E. Bazhenova in co-authorship with S. Buelow and M. Weske, published in [23].

The rest of the chapter is structured as follows. Section 6.1 introduces the chapter by providing a motivating example demonstrating how process decision logic can be incorporated in an event log. Section 6.2 outlines our proposed methodology to extract decision models from event logs, complementary to processes, and discusses assumptions that we rely on, which is based on the contribution of E. Bazhenova to the results published in [21] and [23]. In Section 6.3, we distinguish between different types of decisions that can be detected in an event log and propose an algorithm on their extraction from the event log, which is based on the contribution of E. Bazhenova to the results published in [21] and [23]. Section 6.4 introduces our algorithm to discover dependencies between extracted decisions. In Section 6.5 we validate the presented methodology by applying it to a simulated event log. Sections 6.4–6.5 is based on a joint work of all co-authors from [21] and [23]. The chapter is concluded by a discussion of results.

6.1 MOTIVATING EXAMPLE

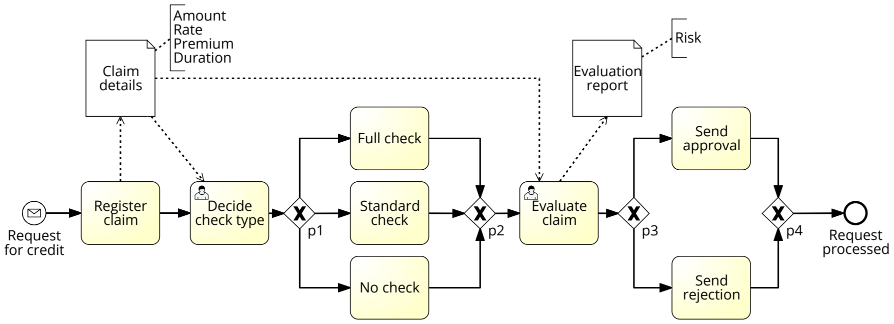


Figure 39: Process model of the loan application in a bank

In order to illustrate how decision logic of processes can be incorporated in event logs, in this section we provide a motivating use case. Consider an example process representing a loan application in a bank, as shown in Figure 39. The process is triggered when the *Request for credit* message event arrives. Further, the *Registration claim* activity is executed. *Claim details* are recorded in the bank system as the attributes of the activity output data node: the claimed *Amount* (e.g., in EUR), the desired payback *Rate* (e.g., in EUR) per month, the payback *Duration* (e.g., in months), and if the customer has a *Premium* status. Afterwards, an expert executes the *Decide check type* activity, in order to determine what type of check should be executed next. Thereby, when the corresponding type of check is determined, the activities *Full check*, *Standard check*, or *No check* represent recording of the corresponding decision in the bank system by the process expert. That said, the activities *Full check* and *Standard check* also represent the actual checks executed for the claim. The process proceeds with execution of the *Evaluate claim* activity, whereby the expert decides if the client's claim is accepted or rejected. Accordingly, the claim *Risk* is recorded as the attribute of the *Evaluation report* data node. After sending an approval, or a rejection letter, the request is considered processed.

Once the example business process is designed and configured, it can be executed with the help of a process-aware information system (PAIS). Thereby, an event log (cf. Definition 17) is a central artifact of process execution.

Table 8 shows an example fragment of the event log for the process depicted in Figure 39. For each event instance e, the event log records the event ID, the trace ID referring to the corresponding process instance, the name of the executed activity, and the set of other event attributes $a \in A$ logged when the corresponding activity writes the attributes of the output data node. For example, the event instance 1 has the following attributes: *Amount* $a_1 = 84$ [EUR], *Rate* $a_2 = 2.8$ [%], *Duration* $a_3 = 30$ [Mths], *Premium* $a_4 = false$. All other information, e.g., the timestamps of event instances is discarded.

Event ID	Trace ID	Name	Other attributes
1	1	Register claim	Amount = 84 [EUR], Rate = 2.8 [%], Duration = 30 [Mths], Premium = false
2	1	Full check	-
3	2	Register claim	Amount = 80 [EUR], Rate = 4.4 [%], Duration = 18 [Mths], Premium = true
4	2	No check	-
5	1	Evaluate claim	Risk = 3
6	1	Send rejection	-

Table 8: An excerpt of the event log for the process depicted in Figure 39

The main problem with managing such decision making is that the decisions done by experts are only implicitly contained in the event log. For example, an expert from a bank compliance department would not be able to identify if the logic behind the decision recorded in the event log (see Table 8) is non-discriminatory, since the decision rules are not logged anywhere. The existing techniques on rules mining [205] allow the knowledge about the process decisions to be empirically derived from the logged expert decisions depicted in Table 8 in the form of credit evaluation rules. However, the corresponding process model depicted in Figure 39 does not allow for decision knowledge to be obtained. Moreover, direct application of the empirically derived rules for the development of credit scoring systems can lead to the unjust treatment of an individual applicant, e.g., judging the applicant's creditability by the first letter of a person's last name [37]. Thus, it seems reasonable to use automated credit scoring systems [137] complemented with an explanatory model. Therefore, we propose further in this chapter a methodology to derive a decision model from the process event log, using DMN for decision modeling. An advantage of utilizing a DMN decision model complementary to the business process model from Figure 39 is that it supports the separation of process and decision logic principle described by us in Section 2.3.3.

6.2 METHODOLOGY AND ASSUMPTIONS

Existing approaches for discovery of decision rules *in the business process context* concentrate on the retrieval of control flow decisions but neglect data decisions and dependencies that are contained within the logged data (for details, see the related work presented in Section 3.5.2). To overcome this gap, we extended an existing approach to derive control flow decisions from event logs [165] with additional identification of data decisions and dependencies between them. Furthermore, we proposed an algorithm for detecting dependencies between discovered control flow and data decisions. The result of the application of our methodology is a complete DMN decision model which explains the executed decisions, and can serve as a blueprint for further decision management.

Figure 40 visualizes the four steps of our methodology and the corresponding input and output. Given an event log (cf. Definition 17), we first detect process decisions based on the specifications of Section 6.3. The detected decisions should be mapped to the corresponding elements of a decision model, in adherence to a technique presented in Section 6.3.3. Next, the dependencies between discovered decisions should be detected according to techniques presented in Section 6.4. The detected dependencies should also be mapped to the corresponding elements of a decision model, in accordance to a technique presented in Section 6.4.3. The output of the methodology is a decision model which explains "as-is" decision making in the given event log.

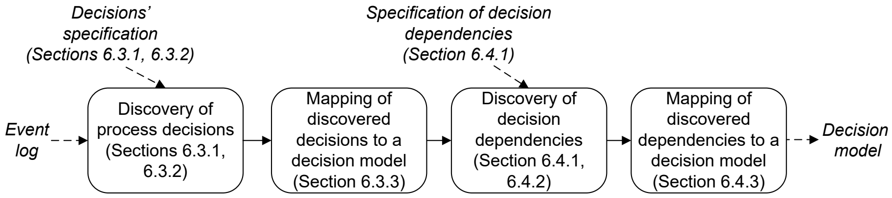

Decisions' specification (Sections 6.3.1, 6.3.2)
Specification of decision dependencies (Section 6.4.1)
Event log
Discovery of process decisions (Sections 6.3.1, 6.3.2)
Mapping of discovered decisions to a decision model (Section 6.3.3)
Discovery of decision dependencies (Section 6.4.1, 6.4.2)
Mapping of discovered dependencies to a decision model (Section 6.4.3)
Decision model

Figure 40: An outline of our methodology to discover decision models from event logs presented in the rest of Chapter 6

In addition to the techniques presented in this chapter, a process model can be extracted from the event log. Thereby, both process and decision models can be used in a complementary way which realizes the separation of concerns principle (cf. Section 2.3.3). There exists a well-established process mining discipline which provides means for automated discovery of process-related information from event logs created by IT systems [63, 184]. Thereby, the process mining dicsovery is out of the scope of this thesis, but we assume that the corresponding process model can be extracted from a given event log, and output process models can be used complementary to decision models discovered from the log according to the techniques developed by us.

For extraction of a decision model from a given event log, we rely on several general assumptions as follows:

1. Historical process information (i.e., the start or end of activities) is available as an event log (cf. Definition 17). Additionally, a process model is available, which can be extracted

from the event log by any process mining algorithm, e.g., from [32, 55, 57, 113, 184, 187].

2. Given event log corresponds to a process model which is structurally sound, i. e., *pm* contains exactly one start and one end event and every node of *pm* is on a path from the start to the end event. This assumption is reasonable, as usually models exhibiting deadlocks, or livelocks are assumed to be subject to modeling errors [61].

3. If the input event log encodes execution of processes incorporating split gateways, the decisions for them are assumed to be taken in distinct process model activities preceding the split gateways. It is considered as a mistake made by many inexperienced BPMN users to omit the decision activity preceding the split gateway [197].

4. Further, we assume that the process decisions do not appear within loops. This would result in several instances of one decision within one process execution, which is a known obstacle for classifying decisions in event logs of processes [165]. For example, an event log obtained by execution of the process model from Figure 41 may contain multiple occurrences of activities B and C associated with one process instance. However, only the first occurrence of either of them should be related to the control-based decision made in gateway $p1$. On the other hand, the occurrences of activities B and C can be related to the control-based decision $p4$. Since the classification problem can not be fully specified for such situations in a general way, these cases require a special handling. We leave this handling out of the scope of the current thesis, but we provide a discussion of the possible workarounds on how to preprocess an event log containing the looping decisions in the chapter conclusion. These workarounds are demonstrated when we apply the devised methodology on a real-life event log in Chapter 8.

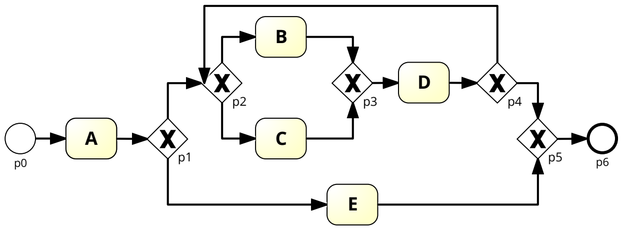


Figure 41: A process model involving decision looping

5. For discovering the decision models from event logs, as presented in Section 6.2, we solve the classification problem where the classes are process decisions, and the training examples are

the process instances recorded in the event log. Hereby, in the presented algorithms, we assume that the classification correctness is such that outcome decision models consists of elements that classify *all* training instances correctly. Thereby, in real-life applications, the classification correctness can be adapted for business needs by using a user-defined correctness threshold in percent. We provide experiments with a user-determined threshold in Chapter 8.

Relying on the presented assumptions, in the next sections we present the steps of our methodology to discover decision models from event logs in accordance to the outline presented in Figure 40.

6.3 DISCOVERY OF DECISIONS FROM EVENT LOGS

In this section, we introduce different types of decisions that can be detected in a given event log. Section 6.3.1 presents control flow decisions, and Section 6.3.2 presents decisions over the data attributes from the event log, which we call for simplicity "data decisions". In both cases, we provide algorithms to detect the decisions in the event log. Further, we provide a technique to map the discovered decisions with the corresponding elements of a decision model, on the example of DMN decision model in Section 6.3.3.

6.3.1 *Control Flow Decisions*

The results of our analysis of how companies design their decision making, presented in Chapter 4, show that the *split gateway* pattern (cf. Definition 28) is the most common pattern for modeling process decisions. With respect to the existing works, the notion of a *decision point*, introduced by Rozinat et al. in [165], is closely related to this pattern. In particular, a decision point represents the case of the split gateway pattern which incorporates a gateway of the exclusive split type. When the decision activity from the decision point is executed, the gateway routes the process towards execution of one from the alternative activities, to which we refer as *decision outcomes*. In Figure 39, for example, such decision on process routing occurs when a decision activity *Decide check type* is executed, and gateway $p1$ routes the process towards execution of one from the alternative activities *Full check*, *Standard check*, or *No check*.

We assume that the decision activity, preceding the exclusive split gateway, refers to the same business decision as the gateway (Assumption 3 from Section 6.2). Omission of this decision activity is a common mistake made by inexperienced BPMN users [197]. Under this assumption, further we define the notion of control flow decisions that can be encountered in a given process model. Also, we propose an approach to discover control flow decisions from a given event log. Thereby, the control flow decision defined by us corresponds to the decision point notion which was introduced rather informally in [165].

Given an event log, the corresponding process model can be discovered from it by any process mining algorithm, e.g., from [32, 55, 57, 113, 184, 187]. According to Assumption 1 from Section 6.2, such process mining algorithm is available, so we leave handling the process mining procedure out of the scope of our work.

Given that a process model pm (cf. Definition 7) is discovered from an event log L (cf. Definition 17), we determine constructs of directly succeeding control flow nodes which represent a gateway succeeded by an activity on each of the outgoing paths. The step output is a set of split exclusive gateways of the process model pm. Further, the control flow decision rules for these gateways can be established by analyzing the choices which have been made in past process executions. In particular, we consider that the input expression of a corresponding control flow decision rule can be mapped the values of attributes recorded in the event log, and the rule output can be mapped to decision outcome of the control flow decision. Thus, we formally define a control flow decision in Definition 18 as follows.

Definition 18 (Control Flow Decision). Given is an event log L that has a set of attributes $A = \{A_1, ..., A_v\}, v \in \mathbb{N}^+$, and a corresponding process model pm. Let $P \subseteq G$ be a set of split exclusive gateways of pm, such that $\alpha_g(p) = XOR$ for each $p \in P$. Additionally, let $T_p = \{t_1, ..., t_k\}$, $T_p \subseteq T$, $k \in \mathbb{N}^+$ be a finite non-empty set of activities of pm directly succeeding each gateway $p \in P$, such that for each activity $t \in T_p$ there exists a directed flow edge $(p, t) \in C$. A gateway $p \in P$ is a *control flow decision* if there exists a finite non-empty set of mappings (decision rules) $R = \{R_1, ..., R_n\}$, $n \in \mathbb{N}^+$, which relate a subset of the given set of attributes A to one of the activities $t_i \in T_p, i{\in}\mathbb{N}^+$ directly succeeding this gateway, as follows:

$$\forall i \in [1;n]\ R_i : \bigwedge_{j=1}^{w} (A_j\ op_j\ q_j) \longrightarrow t_i,\ \ 1 \leq w \leq v, j, w \in \mathbb{N}^+ \qquad (3)$$

where the attributes $A_1, ..., A_w$ are decision inputs, $op_1, ..., op_w$ are comparison predicates, $q_1 \in Dom(A_1)$, ... , $q_w \in Dom(A_w)$ are constants representing values from the domains of the attributes, and $t_i \in T_p$, is a decision output. ◇

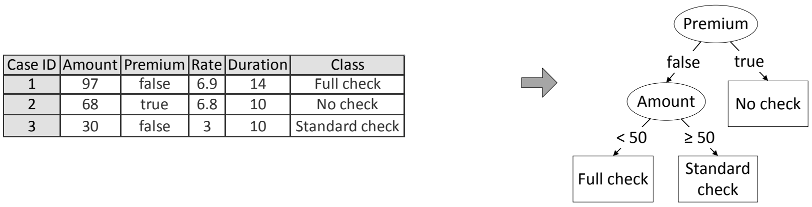
For example, the decision rule for the control flow decision $p1$ (see Figure 39) is:

$$Premium = false,\ Amount < 50 \longrightarrow Full\ check \qquad (4)$$

Thereby, the detected control flow decisions described in Definition 18 correspond to decision nodes in the DMN decision requirements diagram (cf. Definition 12). The decision nodes are especially useful for describing the process decision making when they additionally reference the decision logic level. In our work, we use the decision table notion for expressing the decision rules of processes (cf. Definition 13). Thus, the control flow decision rules from Definition 18 represent a special case of the decision rules from Definition 13, where the rule inputs are data attributes from the event log, and the rule output are decision outcomes of the control flow decision.

The control flow decision rules can be derived using the approach introduced in [165]. It is based on the idea that a control flow decision can be turned into a classification problem, where the classes are process decisions that can be made, and the training examples are the process instances recorded in the event log.

In accordance with [165], we propose to use decision trees for solving the presented problem. Among other popular classification algorithms are neural networks [12] and support vector machines [116]. Pursuing the goal of deriving an *explanatory* decision model for a business process, we utilize the decision tree classification mechanism, as it delivers a computationally inexpensive classification based on few comprehensible business rules with a small need for customization [13]. Also, the decision tree classification algorithms are able to deal with continuous-valued attributes, missing attribute values, and they include effective methods to avoid overfitting the data [205]. Also according to [120], decision tree algorithms are very popular methodologies since the algorithms have a simple inference mechanism and provide a comprehensible way to represent the model in the form of a decision tree. The output decision tree can be straightforwardly transformed into a decision table, since each path in the decision tree represents a decision rule [12].



Case ID	Amount	Premium	Rate	Duration	Class
1	97	false	6.9	14	Full check
2	68	true	6.8	10	No check
3	30	false	3	10	Standard check

(a) Training examples

(b) Decision tree

Figure 42: Control flow decision $p1$ represented as classification problem

To illustrate the classification problem on our running example, we visualized it in Figure 42. The possibly influencing attributes for the control flow decision $p1$ are: *Amount, Premium, Rate, Duration*. The learning instances are created using the information from the event log as presented in the table in Figure 42a, where lines represent process instances, and columns represent possibly influencing attributes. The *Class* column contains the decision outcome for a process instance. An example of the classification of process instances by a decision tree is presented in Figure 42b.

6.3.2 *Data Decisions*

Besides the explicit control flow decisions, process models can contain implicit data decisions. By *data decisions* we understand such process decision making when values of the atoms in the decision rules can be determined for a certain process instance by knowledge of the variable assignments of other attributes. For example, in Figure 39, it is not exhibited how the value of an attribute *Risk* is assigned. However, in practice, it depends on the values of attributes *Amount, Rate, Premium* and *Duration* recorded in the system while registering the client's claim. We distinguish the data decisions into functional and rule-based data decisions.

Definition 19 (Rule-Based Data Decision). Given is an event log L that has a set of attributes $A = \{A_1, ..., A_v\}, v \in \mathbb{N}^+$. An attribute A_k, $1 \le k \le v$, $k, v \in \mathbb{N}^+$ is a *rule-based decision* if there exists a finite non-empty set of mappings (decision rules) $R = \{R_1, ..., R_n\}$, $n \in \mathbb{N}^+$ which relate a subset of the given set of attributes A to the aforementioned attribute A_k, as follows:

$$\forall i \in [1;n]\ R_i : \bigwedge_{j=1}^{w} (A_j\ op_j\ q_j) \longrightarrow Dom(A_k),\ 1 \le w \le v,\ j, w \in \mathbb{N}^+ \quad (5)$$

where the attributes $A_1, ..., A_w$ are decision inputs, $op_1, ..., op_w$ are comparison predicates, $q_1 \in Dom(A_1)$, ... , $q_w \in Dom(A_w)$ are constants, and the attribute A_k, is a decision output. ◇

For example, it might be established based on executions of process model from Figure 39 that the attribute *Risk* is a rule-based data decision. An example decision rule for *Risk* in such case would be the following rule:

$$Premium = false,\ Amount < 50,\ Duration < 10 \longrightarrow Risk = 4 \quad (6)$$

Definition 20 (Functional Data Decision). Given is an event log L that has a set of attributes $A = \{A_1, ..., A_v\}, v \in \mathbb{N}^+$. An attribute A_k, $1 \le k \le v$, $k, v \in \mathbb{N}^+$ is a *functional data decision* if there exists a function $f : \{A_1, \ldots, A_w\} \longrightarrow Dom(A_k)$, $1 \le w \le v$, $w \in \mathbb{N}^+$ which relates a subset of the given set of attributes to the aforementioned attribute A_k. ◇

For example, it might be devised based on executions of process model from Figure 39 that the attribute *Duration* is a the following functional data decision:

$$Duration = Amount/Rate \quad (7)$$

In Algorithm 1, we propose a way to retrieve data decisions using a process model *pm*, an event log L, and a corresponding set of attributes of the event instances $A = \{A_1, \ldots, A_v\}, v \in \mathbb{N}^+$ as inputs.

As any of the attributes from the set A can potentially be the ouput of a data decision, the procedure runs for each attribute $a \in A$ (line 2).

Algorithm 1 Discovery of Rule-Based and Functional Data Decisions from an Event Log

```
 1: procedure FindDataDecisions(processModel $pm$, eventLog $L$, attributes $A$)
 2:     for all $a \in A$ do
 3:         $A_{inf} \leftarrow$ possibly influencing attributes for $a$
 4:         $dt \leftarrow$ decision tree for $a$ using $A_{inf}$ as features
 5:         if $dt$ correctly classifies all instances then
 6:             return rule-based data decision for $a$
 7:         else
 8:             if $a$ has a numeric domain then
 9:                 $A_{inf_{num}} \leftarrow$ attributes from $A_{inf}$ with numeric domain
10:                 $operators \leftarrow$ {+, -, *, /}
11:                 $funcs \leftarrow$ {function "$a\ o\ b$" | $a, b \in A_{inf_{num}} \wedge o \in operators$}
12:                 for all $func \in funcs$ do
13:                     if $func$ correctly determines value of $a$ for all instances then
14:                         return functional data decision for $a$
15:                         break
16:                     end if
17:                 end for
18:             end if
19:         end if
20:     end for
21: end procedure
```

Firstly, in line 3, a set A_{inf} of attributes possibly influencing a is determined using the assumption that all the attributes are recorded in the event log before or equal to the transition to which the attribute a is referred. Afterwards, the procedure detects whether an attribute a represents a *rule-based data decision* (lines 4 - 6). For this, we build a decision tree dt classifying the values of the attribute a using the set of possibly influencing attributes A_{inf} as features. If the built decision tree classifies all training instances correctly, a rule-based data decision for the attribute is yielded (we assume that the classification correctness can be adapted for business needs by using a user-defined correctness threshold in percent, cf. Assumption 5 from Section 6.2). If no set of rules was found, the algorithm searches for a *functional data decision* for attribute a (lines 8 - 18). For this, we consider only such attributes which have a numeric domain. We determine the function form by a template representing combinations of two different attributes from A_{inf} connected by an arithmetic operator (10). The functions representing all possible combinations of such kind, are tested for producing the correct output for all known instances (lines 11 - 12). If that is the case, a functional data decision for a is returned (line 14). If it is determined that an attribute a is neither a rule-based, nor a functional data decision, a is discarded as a possible data decision and is treated as a normal attribute.

For our running example, the algorithm finds a rule-based data decision for attribute *Risk* depending on the attributes *Amount*, *Premium*, and *Duration* as presented in Figure 43a. An instance of a functional decision is Equation 7.

6.3.3 *Mapping of the Discovered Decisions with the DMN model*

The detected decisions represent decision nodes in the DMN decision requirements diagram (cf. Definition 12) which conforms to the original process model. The algorithm for constructing this diagram is straightforward. Firstly, for each discovered control flow decision $p \in P$, we create a new decision node $d \in D$ which is added to the set of decision nodes D of the decision requirements diagram. Further, for each attribute of the event log $A_j \in A = \{A_1, \ldots, A_v\}, v \in \mathbb{N}^+$ it is checked whether it is a rule-based or functional decision over other attributes from the set of attributes A. If this is the case, then a new decision node $d \in D$ corresponding to this attribute A_j is added to the output decision requirements diagram DRD. Otherwise, we create a data node $id \in ID$ corresponding to this attribute A_j which is added to the set of input data nodes ID of the decision requirements diagram. The decision nodes reference the corresponding decision tables containing the extracted rules. An example mapping is presented in Figure 45a.

The decision layer of the output DMN model is represented by the output decision tables (cf. Definition 13) which we obtain from decision trees discovered in accordance to techniques presented above in this section. Thereby, the decision nodes from the DRD layer are associated with corresponding decision tables from the decision logic layer.

6.4 DISCOVERY OF DECISION DEPENDENCIES FROM EVENT LOGS

The decisions discovered in process models as presented above, are used for creating a set of the decision nodes D in the output decision requirements diagram DRD. For now, these decisions represent isolated nodes in the decision requirements diagram, and to "connect" them by the information requirements IR, in this section we propose an algorithm of discovering the decision dependencies from the event log. Section 6.4.1 presents the base algorithm to detect dependencies between process decisions. This algorithm makes the assumption that a decision $d_1 \in D$ that depends on another decision $d_2 \in D$ cannot additionally depend on attributes from the event log that d_1 depends on. Section 6.4.2 shows how to overcome this assumption and introduces an improved approach for finding dependencies between process decisions.

6.4.1 *Base Algorithm*

We propose to distinguish between two following types of decision dependencies.

Definition 21 (Trivial decision dependency). If a decision d in the set of decision nodes detected in the event log ($d \in D$) depends on the attribute $a_k \in A, k \in \mathbb{N}^+$ from the event log and this attribute is detected to be a data decision ($dd \in D$), then there is a *trivial dependency* between this decision d and the decision dd. ◇

Definition 22 (Non-trivial decision dependency). *Non-trivial decision dependencies* are dependencies between detected decisions of any type and control flow decisions influencing them. Let a decision $d \in D$ be in the set of decision nodes detected in the event log. A control flow decision d_{inf} also detected in the event log ($d_{inf} \in D$) is considered as possibly influencing for decision d if two conditions are satisfied: (1) d_{inf} is bound to a transition in the model that lies before or is equal to the transition of d; and (2) the set of influencing attributes of d_{inf} is a subset of the set of influencing attributes of d. ◇

To find the dependencies between either control flow, or data decisions detected in the event log of a process model, we propose Algorithm 2. The inputs to the algorithm are process model pm, event log L, and a corresponding set D of the detected decisions. Each decision d is tested for being influenced by other decisions (line 2). Firstly, the algorithm searches for *trivial decision dependencies* (lines 3 - 6). In line 3, we identify all attributes A_{inf} influencing the decision d (those are the attributes appearing in the set of rules or in the function of the decision). For each attribute a_{inf} in the set of influencing attributes A_{inf}, we check if there is a data decision deciding a_{inf}. If this is the case, the algorithm yields a trivial decision dependency between this data decision on a_{inf} and the decision d (line 6).
Afterwards, the algorithm searches for *non-trivial decision dependencies* (lines 9 - 14). We firstly identify a set of control flow decisions D_{inf} possibly influencing the decision d in line 9. Data decisions are not considered, because a data decision influencing another decision always results in a trivial decision dependency. Any decision $d_{inf} \in D_{inf}$ should satisfy two conditions: (1) d_{inf} is bound to a transition in the model that lies before or is equal to the transition of d; and (2) the set of influencing attributes of d_{inf} is a subset of the

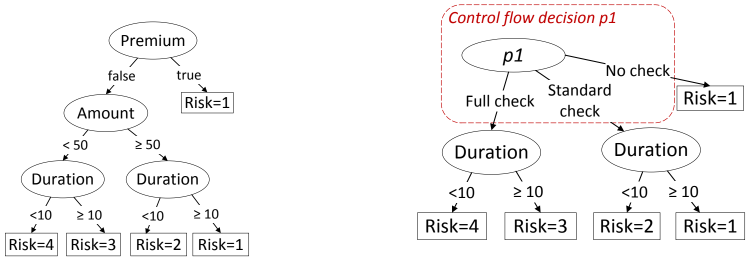
Algorithm 2 Discovery of Decision Dependencies from an Event Log

```
 1: procedure FindDependencies(processModel $pm$, eventLog $L$, decisions $D$)
 2:     for all $d \in D$ do
 3:         $A_{inf} \leftarrow$ attributes influencing $d$
 4:         for all $a_{inf} \in A_{inf}$ do
 5:             if $D$ contains a data decision $dd$ for $a_{inf}$ then
 6:                 return decision dependency $dd \rightarrow d$
                                                  ▷ trivial dependency
 7:             end if
 8:         end for
 9:         $D_{inf} \leftarrow$ possibly influencing control flow decisions for $d$
10:         for all $d_{inf} \in D_{inf}$ do
11:             $Feat \leftarrow A_{inf}$ - attributes influencing $d_{inf} \cup \{d_{inf}\}$
12:             $dt \leftarrow$ decision tree for $d$ using $Feat$ as features
13:             if dt correctly classifies all instances then
14:                 return decision dependency $d_{inf} \rightarrow d$
                                                  ▷ non-trivial dependency
15:             end if
16:         end for
17:     end for
18: end procedure
```

set of influencing attributes of d. Next, the algorithm detects whether there is a non-trivial decision dependency between each found possibly influencing decision d_{inf} and d. For this sake, we determine a new set of features *Feat* for the decision d (line 11). This set consists of the attributes influencing d without the attributes influencing d_{inf}, but with the output of decision d_{inf}. Using the features from *Feat*, we build a new decision tree deciding on d. If the tree is able to correctly classify all training instances, we have found a non-trivial decision dependency between d_{inf} and d.

In our example, the decision on attribute *Risk* depends on the attribute *Duration* (Equation 6). As we identified *Duration* as being a data decision (Equation 7), there is a trivial decision dependency between the decisions *Duration* and *Risk*. For an example of a non-trivial dependency, have again a look at the *Risk* decision in Figure 43b. The found decision tree for *Risk* depends on the attributes *Amount, Premium* and *Duration*. We identify the control flow decision $p1$ as a possibly influencing decision, as (1) the decision $p1$ happens before the decision *Risk*; and (2) its influencing attributes (*Amount, Premium*) are a subset of the influencing attributes of the decision *Risk*. Further, we can build a decision tree that correctly classifies all instances by using the output of decision $p1$ instead of the attributes *Amount* and

Premium. Note that the attribute *Duration* is in the output decision tree in Figure 43b, as it is not part of the decision $p1$.



(a) Rule-based data decision *Risk*

(b) Decision dependency between $p1$ and *Risk*

Figure 43: Decision trees for attribute *Risk*

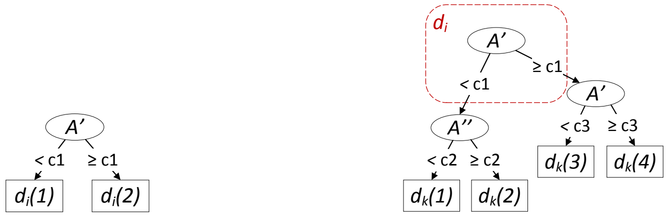
It might be the case that circular dependencies between attributes in the same transition in process model are discovered. For example, in Equation 7, the discovered functional data decision *Duration* depends on *Amount* and *Rate*. However, as *Duration*, *Amount* and *Rate* appear in the same transition, Algorithm 2 finds the data decisions for *Amount* depending on *Duration* and *Rate*, as well as for *Rate* depending on *Duration* and *Amount*. However, in the output decision model two of these three data decisions should be discarded to avoid cyclic dependencies. We leave it to the process expert to determine which decision out of a set of cyclic data decisions is the most relevant for the output decision model.

6.4.2 *An Improved Approach to Find Non-Trivial Decision Dependencies*

The decision dependencies in Algorithm 2 are retrieved under the assumption that if an arbitrary decision d_k depends on another decision d_i ($k, i \in \mathbb{N}^+$), then this decision d_k can *not* additionally depend on attributes that d_i depends on. This problem is illustrated in Figure 44: here, d_i depends on attribute A' and d_k depends on decision d_i and additionally on the attributes A' and A''. The Algorithm 2 tries to rebuild the decision tree for d_k without considering the attributes of d_i, and it only utilizes attribute A''. Thus, this algorithm finds no dependency between d_i and d_k, which is not correct.
To overcome this problem, we propose an alternative Algorithm 3 for finding non-trivial decision dependencies in the process event log (finding of trivial decision dependencies is equivalent to lines 3 to 8 in Algorithm 2). Firstly, the Algorithm 3 identifies a set of possibly influencing control flow decisions D_{inf} and for each $d_{inf} \in D_{inf}$ it builds a decision tree dt_{inf} containing (1) one root node, that splits according to d_{inf}; (2) as many leaf nodes as d_{inf} has decision outcomes, thereby, each of them containing a set of learning instances. Then, for each leaf node a subtree is built that decides on d for all learning

Algorithm 3 Improved Retrieving of Non-Trivial Decision Dependencies

```
procedure FindDependenciesNew(processModel m, eventLog L, decisions D)
    for all $d \in D$ do
        $D_{inf} \leftarrow$ possibly influencing control flow decisions for $d$
        for all $d_{inf} \in D_{inf}$ do
            $dt_{inf} \leftarrow$ decision tree where the root node splits
                    according to $d_{inf}$
            $dt_{new} \leftarrow dt_{inf}$
            for all $leaf \in dt_{inf}$ do
                $dt_{leaf} \leftarrow$ decision tree for $d$ classifying instances
                        from $leaf$
                $dt_{new} \leftarrow$ add $dt_{leaf}$ to the $leaf$ of $dt_{new}$
            end for
            $levels_{inf} \leftarrow$ number of levels of $dt_{new}$
            $levels_{orig} \leftarrow$number of levels of original decision tree
                    for $d$
            if ($levels_{inf} <= levels_{orig}$ then
                return decision dependency $d_{inf} \rightarrow d$
            end if
        end for
    end for
end procedure
```



(a) The decision tree for decision d_i (b) The decision tree for decision d_k

Figure 44: Decision d_k depends on another decision d_i, *and* on the attributes influencing d_i

instances of this leaf, thereby the features are all attributes that were used in the originally found decision tree for d. In lines 7, all subtrees are attached to dt_{inf} resulting in dt_{new} which is a decision tree for d. Next, it is checked that the complexity of the newly constructed tree dt_{new} has not increased in comparison to the original decision tree for d by measuring and comparing the corresponding maximum number of nodes from the root to the leafs (lines 11 - 14). If this is the case, then the algorithm outputs a decision dependency between d_{inf} and d.

6.4.3 *Mapping of Discovered Decision Dependencies with DMN model*

In Section 6.3.3, we presented how the decisions discovered in process models can be mapped to the corresponding elements of the DMN model. Thereby, at this step of our methodology, there should exist: (1) a set D of decision nodes in the DRD diagram; (2) a set ID of input data nodes in the DRD diagram; and (3) a set of decision tables corresponding to the decision nodes in the DRD diagram.

Thereby, at this step of our methodology, the trivial and non-trivial decision dependencies detected in the process event log are directly mapped to the set of information requirements IR represented by directed arrows between the discovered decision nodes D and input data nodes ID in the output decision requirements diagram DRD. Herewith, if new classification decision trees are identified by Algorithm 3, the updated decision tables are recorded at the decision logic layer correspondingly. An example mapping is presented in Figure 45b.

Since our methodology for discovery of decision models from data recorded in event logs is based on the decision tree classification, the output decision tables consist of non-overlapping rules that are assigned with the unique hit policies (cf. Definition 13). As discussed in the related work (cf. 3.5.2), there also exist an approach which deals with discovery of overlapping decision rules [122] for standalone decisions. However, future investigations could be done in order to discover decision models which contain interconnected decision tables that consist of overlapping decisions

6.5 APPLICATION OF THE METHODOLOGY ON THE EXAMPLE LOG

To validate our methodology of the decision model discovery from an event log of a process model, we implemented it. In the following section, we discuss application of our methodology to the test example, represented by a process model in Section 6.1 (see Figure 39). The details of our implementation, and the evaluation of the methodology by applying it on a real log can be found in Chapter 8.

For analysing the test process, we created an event log with the help of the simulation system CPN Tools[1]. To create the input for the methodology, we needed to obtain an event log for the process model from Figure 39. Estimating the distributions of parameters from a real credit-risk data set [12], we simulated the event log using the simulation parameters presented in Table 9.

Task / Attribute Name	Simulation Parameters
Trace ID	1 to 200 (incrementing)
Amount	discrete(2,99)
Premium	random boolean
Duration	discrete(2,30)
Rate	*Amount* / *Duration*

Risk	if $Amount \geq 50$ and $Duration > 15$: $Risk = 4$
	if $Amount \geq 50$ and $Duration < 15$ and $Duration > 5$: $Risk = 3$
	if $Amount \geq 50$ and $Duration < 5$: $Risk = 2$
	if $Amount < 50$ and $Duration > 20$: $Risk = 3$
	if $Amount < 50$ and $Duration < 20$ and $Duration > 10$: $Risk = 2$
	if $Amount < 50$ and $Duration < 10$: $Risk = 1$

p1	if $Amount \geq 50$ and $Premium = false$: *Full check*
	if $Amount < 50$ and $Premium = false$: *Standard check*
	if $Premium = true$: *No check*

p3	if $Risk \leq 2$: *Send approval*
	if $Risk > 2$: *Send rejection*

Table 9: Simulation parameters for generating the event log of the process from Figure 39

Further, we discovered control flow decisions from the simulated event log, according to the technique presented in Section 6.3.1. Afterwards, all data decisions were discovered from the log according to our algorithm presented in Section 6.3.2. The validation experiment was concluded by discovery of decision dependencies in accordance to our algorithm presented in Section 6.4. The results of execution of these steps are presented below.

[1] http://cpntools.org/

Discovery of decisions

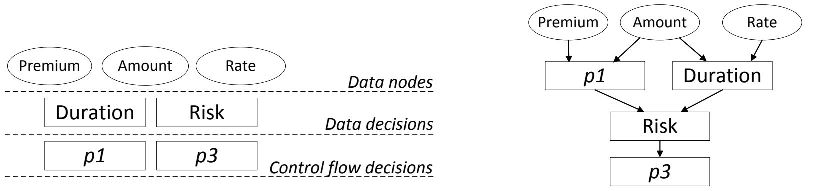
Assuming that only the input event log is available, we mine the process model using the Prom software[2] for process mining. Further, according to approach from Section 6.3.1, we identify two control flow decisions: (1) $p1$ with decision alternatives *Full check, Standard check*, and *No check*; and (2) $p3$ with decision alternatives *Send approval* and *Send rejection*. An example decision tree constructed for $p1$ is depicted in Figure 42. Executing Algorithm 1, we find: (1) A rule-based decision *Risk* (Figure 43a); (2) A functional decision *Duration* (Equation 7).

The aggregate of the decisions discovered by the program is presented schematically in Figure 45a. Those are the elements which are used further for the construction of the decision requirements diagram: (1) Data nodes (*Premium, Amount, Rate*); (2) Data decisions (*Duration, Risk*); and (3) Control flow decisions ($p1$, $p3$).

Discovery of decision dependencies

Further, we execute Algorithm 2 to mine the dependencies between the discovered decisions from Figure 45a, and it outputs the fully specified DMN decision model as depicted in Figure 45b. Thus, the program finds the trivial dependencies between the decisions *Duration* and *Risk*, as well as between *Risk* and $p3$, also, the non-trivial dependency between the decisions $p1$ and *Risk*. In case of circular dependencies, a random decision is kept.

The extracted decision model (Figure 45b) shows explicitly the decisions corresponding to the process from Figure 39, and thus, could serve for compliance checks by explaining the taken decisions. Also, the derived decision model can be executed complementary to the process model, thereby supporting the principle of separation of concerns [197].



(a) Discovered decisions

(b) Discovered decision requirements diagram

Figure 45: The discovered decisions and the DMN model for the example process

[2] http://www.promtools.org/

6.6 SUMMARY AND DISCUSSION

In this chapter, we provide a methodology enabling the extraction of decision models from event logs, on the example of DMN. Firstly, we present the procedures to identify decisions in a process model. Secondly, we present the extraction of the decision logic containing the data dependencies derived from the event log. Thereby, we provide the algorithms to construct a fully specified DMN decision model.

Our methodology is an extension of the existing approach to derive control flow decisions from event logs with additional identification of data decisions and dependencies between them. Thus, data decisions are discovered from the input event log with the help of decision tree classification in case of rule-based data decisions, or with a template-based approach in case of functional data decisions. Furthermore, we proposed a modified approach to rebuild decision trees to identify the dependencies between discovered decisions and overcame the problem of reusing attributes in a dependent decision.

The extracted DMN decision model reflects the decisions detected in the event log of a process model, which could be served as an explanatory model used for compliance checks. Additionally, executing this model complementary to the process model supports the principle of separation of concerns by providing increased flexibility, as changes in the decision model can be executed without changing the process model.

One of the limitations of the presented methodology is that the decisions do not appear within loops. Whereas we do not provide a formal approach for handling the looping decisions, in a real-world setting, the following heuristic can be applied. In particular, the input event log can be preprocessed in such a way that process instances that contain several decisions with respect to the current decision point, should be split up. In such a way, the looping is avoided, and these different process instances serve as multiple learning instances.

We validated our methodology by applying it to a test event log. The results show that the methodology yields a comprehensible decision model which can be used as complementary to the process model obtained by the process mining algorithms. The event log was simulated by taking into account limitations stated in Section 6.2. Moreover, in the presented algorithms of discovery decisions and their dependencies, we used the requirement of correct classification of all the instances. The extended evaluation of presented methodology is presented further in Chapter 8. In particular, we apply the introduced methodology to discover decision models to a real-life event log, and by that we explore its advantages and limitations in a real-life context.

7 DISCOVERY OF FUZZY DECISION MODELS FROM EVENT LOGS

The closer one looks at a real-world problem, the fuzzier becomes its solution.
— Lofti A. Zadeh, Creator of Fuzzy Logic

In the previous chapters, we addressed discovery of decision models complementary to process models from event logs, whereby the decision logic is represented by decision rules based on Boolean algebra. The formal nature of such decision rules is often hard for interpretation and utilization in practice because imprecision is intrinsic to real-life decisions [193]. Operations research considers fuzzy logic, based on fuzzy algebra, as a tool dealing with partial knowledge [27, 209]. In this chapter, we explore the possibility of incorporating fuzziness into DMN decision models. Further, we propose a methodology for discovering fuzzy DMN decision models from event logs. Figure 46 outlines the input and the output of the proposed methodology.

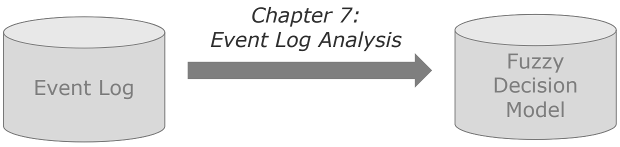


Figure 46: Input and output of the methodology presented in Chapter 7

The proposed methodology to discover fuzzy DMN decision models from event logs is mostly based on a joint work of E. Bazhenova in coauthorship with S. Haarmann, S. Ihde, A. Solti, and M. Weske, which was published in [24]. Additionally, the proposed methodology incorporates discovery of crisp decision requirements diagrams, which is based on results published by Bazhenova et al. in [21] and [23] and which is discussed in Chapter 6.

The rest of the chapter is structured as follows. Section 7.1 introduces the motivation for deriving fuzzy DMN models on an example. Section 7.2 presents our formal framework describing the incorporation of fuzziness in decision models, which is based on results contributed by E. Bazhenova to [24]. Section 7.3 introduces the methodology to discover fuzzy decision models from event logs, which is based on a joint work of E. Bazhenova and all co-authors from [21], [23], and [24]. We validate the presented methodology on the motivating example in Section 7.4, which is based on results of a joint work of E. Bazhenova in co-authorship with S. Haarmann and S. Ihde, which was published in [24]. The chapter is summarized in Section 7.5.

7.1 MOTIVATING EXAMPLE

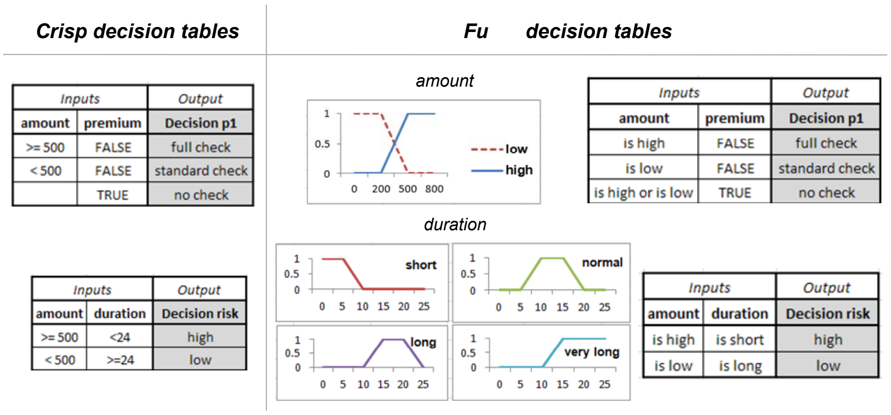
To assist enterprises with efficient decision management, knowledge about "as-is" decision making needs to be retrieved. In Chapter 6, we proposed a methodology to achieve this by analyzing process event logs and discovering decision rules from them. Thereby, we assumed that decision logic is represented by *crisp* decision rules based on Boolean algebra (cf. Definition 16). However, application of this concept is limited in the real world, because humans often deal with concepts which do not have precisely defined boundaries. For example, the natural language concepts (facts) such as *many, very short, much heavier than, quite old*, etc. are true only to some degree and they are false to some degree as well. The fuzzy logic is a mathematical tool created for dealing with imprecision and information granularity which was introduced by Zadeh in [209] and has been widely applied in different disciplines [27].

Fuzzy rules represent strings encoding the semantic meaning of a certain probability behind a value range, e.g., "If loan duration is long, then risk is very high". Since the meaning can be derived directly from the representation, it is generally considered that fuzzy rules are more comprehensible compared to crisp rules [89]. Moreover, using literals across rules increases decision flexibility, as it allows a consistent adaptation of all rules by adjusting only the underlying mappings.

Event ID	Trace ID	Name	Other attributes
1	1	register claim	duration = 14 [Mths], amount = 597 [EUR], premium = false
2	1	full check	-
3	2	register claim	duration = 28 [Mths], amount = 200 [EUR], premium = true
4	2	no check	-
5	1	evaluate	risk = high
6	1	send rejection	-

Table 10: An excerpt of the example event log which serves as an input for discovery of fuzzy decision models. The attribute values of the event log contain both numerical and nominal values.

To demonstrate how fuzzy logic can be utilized for designing decision models complementary to process models, we again consider the example business process of credit-risk assessment from Section 6 (see Figure 39). For this chapter, we modified the example by: (1) considering only three attributes that are recorded in the bank system in the *Claim details* data node – *duration, amount*, and *premium*; (2) consid-



Crisp decision tables

Inputs		Output
amount	premium	Decision p1
>= 500	FALSE	full check
< 500	FALSE	standard check
	TRUE	no check

Inputs		Output
amount	duration	Decision risk
>= 500	<24	high
< 500	>=24	low

Fu decision tables

Inputs		Output
amount	premium	Decision p1
is high	FALSE	full check
is low	FALSE	standard check
is high or is low	TRUE	no check

Inputs		Output
amount	duration	Decision risk
is high	is short	high
is low	is long	low

Figure 47: Two possible types of decision tables corresponding to the decision logic of the example process

ering previously numerical attribute *risk* as nominal with the possible values *very high*, *high*, *normal* or *low*. Hereby, there is an equal amount of numerical and nominal attributes, the values for which are stored in the event log. In the case presented in the current chapter we use the names for activities and attributes that start with the lowercase letters.

The process steps are the same as in Figure 39. The process starts with the registration of the user's claim details such as *duration*, *amount*, and *premium* in the bank information system. At p_1 the type of application check is chosen, and at p_3 the claim is evaluated for approval which happens alongside with assigning *risk* to the claim. An excerpt of the example event log is given in Table 10. The main motivation for discovery of fuzzy decision models is similar to the crisp case: the event log stores only the results of decision making, but it does not explicitly contain the decision logic behind this result. To obtain an explanatory decision model, we propose to derive DMN decision model from the event log.

The DMN models consist of the decision requirements diagram (DRD) and the decision logic layers. The DRD is a model consisting of a set of elements and their interdependencies (cf. Definition 12), thereby, it is crisp by nature. However, the decision logic corresponding to the DRD can be designed in the form of either crisp, or fuzzy decision tables, as shown in the Figure 47. As mentioned earlier, fuzzy rules are considered to be more comprehensible, as a user has an intuitive understanding of the linguistic concepts to which fuzzy rules refer. An example of such mappings are visualized in Figure 47: (1) a trapezoidal membership function representing the membership grade of attribute *amount* in fuzzy sets low, and high; and (2) a trapezoidal membership function representing the membership grade of attribute *duration* in fuzzy sets *short*, *long*, *normal*, and *very long*.

An additional motivation for utilizing fuzzy decision models is that they are highly flexible, since it is possible to execute changes in rules only by adjusting the membership functions. For instance, as it can be seen from the decision rules in Figure 47, the range values of *amount* higher than 500 [EUR] are considered as high. Imagine that the business environment changes, and the boundary value of *amount* should be reconsidered as equal to 800 [EUR]. In the case of crisp decision tables that do not have corresponding membership functions, it would be necessary to change the input values for *amount* four times in corresponding rules. However, in the case of fuzzy decision tables, only the membership function has to be adjusted once.

As discussed in the related work about the mining of fuzzy decision models (cf. Section 3.5), the question of deriving fuzzy decision models complementary to process models from event logs has not received much attention yet. In this chapter, we address this gap by proposing a methodology to derive fuzzy DMN decision models from event logs. The methodology is based on our formal framework, which we present in the next section.

7.2 DEFINITIONS OF FUZZY DECISION CONCEPTS

Facilitation of fuzzy logic implies that the crisp values of data attributes from event logs are to be perceived as a matter of truth values ranging between completely true and completely false [27]. In order to facilitate handling of decision rules discovered from an event log (cf. Definition 17) in a fuzzy manner, below we introduce a set of necessary definitions.

Definition 23 (Fuzzy Subset, Membership Function). Given is an event log L that has a set of attributes $A = \{A_1, \dots, A_v\}, v \in \mathbb{N}^+$. A domain set $Dom(a)$ for an attribute $a \in A$ has K^a *fuzzy subsets* characterized by tuples $FS_i^a = \{(y\ , l_i^a, \mu_{l_i}^a) | y \in Dom(a)\}$, if for each $i \in [1; K^a]$ there exist:

- linguistic terms l_i^a labelling fuzzy subsets FS_i^a;
- *membership functions* $\mu_{l_i}^a : Dom(a) \longrightarrow [0, 1]$ which represent the grade of membership of attribute values from $Dom(a)$ in a fuzzy subset FS_i^a by mapping each value $y \in Dom(a)$ to a real number in the interval $[0, 1]$. ◇

We identify a fuzzy set with its membership function. An example of a membership function is presented with the help of a graph in Figure 47 for the attribute *amount*. In this case, K^{amount}=2, and the domain set $Dom(amount)$ has two fuzzy subsets FS_1^{amount} and FS_2^{amount} characterized correspondingly by membership functions μ_{low}^{amount} and μ_{high}^{amount}. This example is illustrated with the help of *trapezoidal membership function*, as one of the most widely used types of membership functions.

Definition 24 (Trapezoidal Membership Function). The trapezoidal membership function is a function ϕ of an attribute $a \in A$ from the given event log which depends on four scalar parameters y_1, y_2, y_3, and y_4, whereby y_1, y_2, y_3, y_4, are real numbers such that $y_1 \leq y_2 \leq y_3 \leq y_4$, and for any $y \in Dom(a)$ the following holds:

$$\phi(y) = \begin{cases} 0, & y < y_1 \vee y > y_4; \\ \dfrac{y - y_1}{y_2 - y_1}, & y_1 \leq y < y_2; \\ 1, & y_2 \leq y < x_3; \\ \dfrac{y_4 - y}{y_4 - y_3}, & y_3 \leq y < y_4; \end{cases} \tag{8}$$

◇

Nevertheless, the approach proposed in our thesis is general and can be used for the other types of membership functions, e.g., triangular-shaped, Gaussian curve, etc. [27].

Fuzzy logic of decision models can be expressed through fuzzy decision rules and fuzzy decision tables. Fuzzy decision tables consist of fuzzy rules which represent "if-then" mapping between a subset of event log attributes associated with corresponding linguistic terms, and an output process variable associated with a set of labels. We formally define a fuzzy decision table, which consists of elemental fuzzy decision rules, as follows.

Definition 25 (Elemental Fuzzy Decision Rule, Fuzzy Decision Table). Given is an event log L that has a set of attributes $A = \{A_1, \ldots, A_v\}, v \in \mathbb{N}^+$, and given is a finite non-empty set of labels $C = \{C_1, ..., C_z\}, z \in \mathbb{N}^+$. A *fuzzy decision table (FDT)* is a finite non-empty set of mappings (elemental fuzzy decision rules) $R = \{R_1, ..., R_n\}$, $n \in \mathbb{N}^+$, which relate a subset of the given set of attributes A to one of the labels $c_i \in C, i \in \mathbb{N}^+$, as follows:

$$\forall i \in [1; n]\ R_i : \bigwedge_{j=1}^{w} (A_i\ is\ l_{x_i}^{A_i}) \longrightarrow c_i,\ 1 \leq w \leq v,\ j, w, z \in \mathbb{N}^+ \tag{9}$$

where the attributes $A_1, \ldots, A_w$ are decision rule inputs, $l_{x_1}^{A_1}, \ldots, l_{x_w}^{A_w}$ are linguistic terms labeling fuzzy subsets associated to the decision rule inputs, such that $1 \leq x_1 \leq K^{A_1}, \ldots, 1 \leq x_w \leq K^{A_w}$, and $c_i \in C, i \in \mathbb{N}^+$ is a decision output. ◇

In Definition 25, the decision output is represented by a label $c_i \in C, i \in \mathbb{N}^+$ which represents a linguistic term labeling a process decision. As we demonstrate later in this chapter, these labels can refer to either a data attribute from the event log, or to an activity of the process model that can be discovered from the event log. For example, there can exist the following elemental fuzzy decision rules for the fuzzy decision table from Figure 47:

$$\textit{amount is high, premium is FALSE} \longrightarrow p_1 = \textit{full check} \tag{10}$$

$$\textit{amount is low, duration is long} \longrightarrow \textit{risk} = \textit{low} \tag{11}$$

The definition of elemental fuzzy rule sets can be extended to *conjunctive normal form of fuzzy rules (CNF)*, where each attribute $A_i, 1{\leq}i{\leq}v$ can be associated with a set of $p{\in}\mathbb{N}^+$ several linguistic terms $\{l^{A_i}_{x_i,1}, ..., l^{A_i}_{x_i,p}\}$ which are joined by a disjunctive operator. Fuzzy rule bases in CNF form are also called Mamdani rules [27]. An example of CNF fuzzy rule is following (see also FDT in Figure 47):

$$\textit{amount is high or low, premium is TRUE} \longrightarrow p_1 = \textit{no check} \tag{12}$$

Mapping a runtime input for a set of fuzzy rules to an output is called *inference*. There exist multiple inference techniques, and the method that is most closely matching the real-world problem should be chosen [27]. We address this question further in the chapter (see Section 7.3.6).

7.3 METHODOLOGY FOR FUZZY DECISION MODEL DISCOVERY

This section introduces our methodology to discover fuzzy decision models from event logs. Section 7.3.1 presents the approach and the assumptions made, and outlines the rest of Section 7.3.

7.3.1 *Methodology Outline and Assumptions*

As we show in the related work (cf. Section 3.5.2), existing approaches to decision discovery consider only either mining of independent fuzzy rule sets [89], or crisp decision models [23, 53, 165]. In order to provide useful insights into the discovery of fuzzy DMN decision models from event logs, we propose a methodology which consists of five steps, see Figure 48. Given an input event log, the steps to discover fuzzy DMN decision models from the log are: (1) Identification of fuzzy subsets and corresponding membership functions corresponding to the data attributes from the event log, as discussed in Section 7.3.2; (2) Discovery of process decisions and dependencies between them, as presented in Section 7.3.3; (3) Application of fuzzy learners for fuzzy rules discovery which is presented in Section 7.3.4; (4) Construction of fuzzy decision tables based on discovered rules in accordance to techniques presented in Section 7.3.5; and (5) Identification of fuzzy hit policy for decision tables, as discussed in Section 7.3.6.

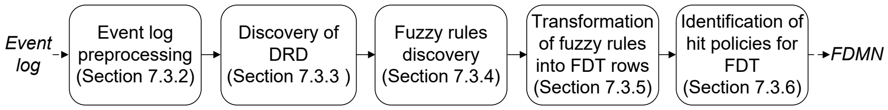

Event log
Event log preprocessing (Section 7.3.2)
Discovery of DRD (Section 7.3.3)
Fuzzy rules discovery (Section 7.3.4)
Transformation of fuzzy rules into FDT rows (Section 7.3.5)
Identification of hit policies for FDT (Section 7.3.6)
FDMN

Figure 48: Our approach for discovering Fuzzy DMN (FDMN) models from event logs

DMN decision model consists of a DRD representing decisions and its dependencies, and of a decision logic layer expressed by sets of rules. As discussed also in Section 7.1, fuzziness is not relevant for the DRD models, because they represent a finite number of decisions that are acts of determining output from inputs, and dependencies between these acts. In contrast, decision rules might incorporate fuzziness through the usage of fuzzy sets of input values in rules. Such a view does not violate the DMN standard, so we adopt it and use the term Fuzzy DMN (FDMN) for such kind of decision models. Thus, the output of our methodology is a FDMN model.

For extraction of a fuzzy decision model from a given event log, we rely on several general assumptions as follows.

1. Historical process information is available as an event log, and additionally, a process model is available which can be extracted from the event log by any process mining algorithm (cf. Assumption 1 in Section 6.2).

2. The given event log corresponds to a process model which is structurally sound (cf. Assumption 2 in Section 6.2).

3. If the input event log encodes execution of processes incorporating split gateways, the decisions for them are assumed to be taken in distinct process model activities preceding the split gateways (cf. Assumption 3 in Section 6.2).

4. In addition to soundness of the process model from the previous assumption, the decisions do not appear within loops (cf. Assumption 4 in Section 6.2).

5. There exists a mechanism that describes numerical attributes of the input event log in linguistic terms, e.g., that the value of an attribute is low or high. This should be done by an expert, as hereby, the domain knowledge is required. Thereby, we provide experts with guidelines on how to increase the automation rate of assigning the log attributes with linguistic terms in Section 7.3.2.

6. For discovering the decision models from event logs, we solve classification problem where the classes are process decisions that can be made, and the training examples are the process

instances recorded in the event log. Hereby, in the presented algorithms, we assume that the classification correctness is such that outcome decision models consists of elements that classify *all* training instances correctly (cf. Assumption 5 in Section 6.2). Thereby, we provide the stakeholders with suggestions for tuning the termination criteria for algoritms in Section 7.3.4.

Note that Assumptions 1–4, and Assumption 6 are similar to the assumptions for the approach to discover crisp decision models from Chapter 6, since the input event log is the same in both cases. Thereby, Assumption 5 refers to the fuzzy extension of mining the decision models from event logs.

Relying on the presented assumptions, we introduce further the steps of our methodology to discover fuzzy decision models from event logs in accordance to the outline presented in Figure 48.

7.3.2 *Event Log Preprocessing*

Given is a an event log L, and a corresponding set of attributes of the event instances $A{=}\{A_1, \ldots, A_v\}, v{\in}\mathbb{N}^+$. Firstly, we preprocess log data in such a way that fuzzy learning of rules can be applied on it. For that, for each attribute there should exist membership functions that describe them in linguistic terms (cf. Definition 23), e.g., that the value of an attribute is "low" or "high". Specifically, we need to find $\mu^a_{l_i} : Dom(a){\longrightarrow}[0, 1], i \in [1; K^a]$ for each attribute $a{\in}A$ from the event log, which we propose to do by the procedure ConstructMF from Algorithm 4.

In the presented procedure, we iterate over all attributes from the event log. If the attribute domain is nominal by itself, the membership function is constructed as a characteristic function (Lines 5–8). Otherwise, if the attribute domain is numerical, an expert should set the number K^a of fuzzy subsets for this domain, that are to be generated, and assign corresponding linguistic terms. For our use case (see Figure 47), the expert would set $K^a{=}2$ for attribute *amount*, and assign two linguistic terms: *low* and *high*.

Function BuildNumAttributesMF can be realized by using two approaches. The first approach involves experts which express their opinions on how well attributes $a{\in}A$ can be associated with fuzzy subsets FS^a_i, $i{\in}[1; K^a]$. The details of constructing the membership functions based on expert opinions are out of scope of this thesis, as they can be found in [211]. The expert approach can be recommended when there is not enough input data allowing to derive the membership function automatically. As the input event log in our case is supposed to be large, we propose to utilize the second approach – the *Fuzzy C-Means (FCM)* algorithm described in [44], as it allows to automatically derive membership functions for data attributes from event logs. Applicable to our case, the FCM-algorithm is able to calculate

Algorithm 4 Preprocessing of Data Attributes from Event Log

```
1:  procedure ConstructMF(eventLog $L$, attributes $A$)
2:      for all $a \in A$ do
3:          if $Dom(a)$ is a set of nominal values then
4:              for all $y \in Dom(a)$ do
5:                  if $y = l$ then
6:                      $\mu_l^a = 1$
7:                  else
8:                      $\mu_l^a = 0$
9:                  end if
10:             end for
11:         else
12:             $K^a \leftarrow$ a number of fuzzy subsets
                                                        ▷ Expert input
13:             for all $i \in [1; K^a]$ do
14:                 $l_i^a \leftarrow$ assign a linguistic term
                                         ▷ Expert input, e.g., "low", "high"
15:                 $\mu_l^a$ =BuildNumAttributesMF()
                                           ▷ Either expert input or FCM
16:             end for
17:         end if
18:     end for
19: end procedure
```

the degree of membership $\mu_{l_i}^a$ of data attribute value a from the event log in each of i clusters, such that $1 \leq i \leq K^a$. FCM requires the number of clusters as an input, which is equal to the number of fuzzy subsets obtained from an expert in Line 12. Example membership functions for our use case are visualized in Figure 47 for variables *amount* and *duration* with respective values.

7.3.3 *Discovery of DRD*

Discovery of DRDs from event logs represents a classification problem over crisp data which was discussed in Chapter 6. Therefore, we present below only a short description of the problem, in order to provide a bridge for the terminologies and notions between Chapter 6 and the current one.

Thus, we distinguish between two types of process decisions: (1) *control flow decisions* represented in process models by split gateways (e.g., decision from Equation 10), and (2) *data decisions* reflecting dependencies between values of data attributes in the event log (e.g., decision from Equation 11). We use the C4.5 classifier for learning both types of decisions by taking event log attributes and transition labels as features, because it generally delivers a computationally inexpensive classification based on few comprehensible business rules, with a small need for customization [205]. For finding dependencies between decisions, we utilize the same classifier, taking the event log attributes and found decision outcomes as features. Each mined decision is added to the set of decision nodes D_{dm} of the output DRD. Each attribute influencing these decisions which is not a decision by itself is added to the set of input data nodes ID of the output DRD. All dependencies are added to the set of information requirements IR of the output DRD.

The output of this step is a DRD (see Figure 47 for an example) consisting of a set of $PDN{\in}\mathbb{N}^{+}$ decisions $\{(A_1, C_1), ..., (A_{PDN}, C_{PDN})\}$, where PDN stands for process decisions number. Herewith, $A_{pdn} = \{A_1, ..., A_g\}{\subseteq}A,\ 1{\leq}g{\leq}v, pdn \in [1, ..., PDN]$, are attributes influencing these decisions. The more detailed description of mining DRDs from event logs is presented and discussed in Chapter 6.

7.3.4 *Fuzzy Rules Discovery*

After discovering process decisions $(A_{pdn}, C_{pdn}),\ 1{\leq}pdn{\leq}PDN$ from the event log, we aim at the discovery of fuzzy rules corresponding to these decisions. Thus, we iterate over the set of PDN decisions and solve for each of them the classification problem presented in Figure 49. Here the training data is comprised in the event log subset $L_{pdn}{\subseteq}L : A_{pdn}{\subseteq}A$ containing only values of attributes that are influencing the given process decision.

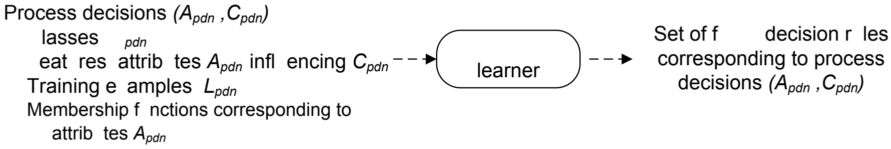

Process decisions (A_{pdn}, C_{pdn})
lasses $_{pdn}$
eat res attrib tes A_{pdn} infl encing C_{pdn}
Training e amples L_{pdn}
Membership f nctions corresponding to attrib tes A_{pdn}
learner
Set of f decision r les corresponding to process decisions (A_{pdn}, C_{pdn})

Figure 49: Classification problem of discovering fuzzy decision rules from event logs

As our goal is to output the explanatory models describing decisions made in the past and recorded in the event log, we take into account that the fuzzy learners should provide good *accuracy* and *interpretability* of results. To explore the appropriateness of known fuzzy classification algorithms in achieving our goals, we did experiments on applying the genetic [89] and NEFCLASS [133] algorithms, both of which are well-known fuzzy classifiers that infer fuzzy classification rules. Below we provide short descriptions of algorithms taking into account modifications needed for solving our problem of deriving fuzzy rule sets from event logs.

7.3.4.1 *Genetic Algorithm (GA)*

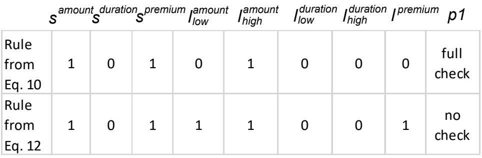
Genetic algorithm (GA) is successfully applied for solving fuzzy classification tasks [44, 79, 89]. Also, GA suits us because it is applicable for training sets with large dimensions, which is typical for event logs. The heuristic character of GA does not guarantee optimality of solution, but it provides approximate solutions close to optimal, which is appropriate for business environments. Below we consider the adaptation of GA applicable to our problem.

GA0. For utilizing GA, we firstly need to represent our problem in a genetic form. For that, we view fuzzy decision rules (FDR) discovered from event logs as so called chromosomes. Given is a process decision (A_{pdn}, C_{pdn}) where $A_{pdn} = \{A_1, ..., A_g\} \subseteq A, 1 \leq g \leq v$ is a set of influencing attributes from the corresponding event log subset L_{pdn}.

Definition 26 (Chromosome). A *chromosome* corresponding to the event log subset L_{pdn} is a string G_t, $t \in \mathbb{N}^+$, which is a concatenation of two bit strings S_t and B_t, where:

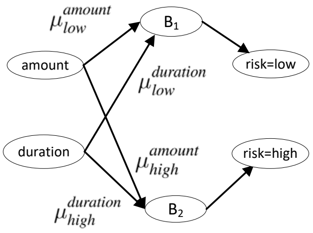
- $S_t = \{s_1, ..., s_v\}$ is a string of bits $s_i, 1 \leq i \leq g$ indicating the presence of an attribute $a \in A_{pdn}$ in a rule antecedent of the chromosome;
- $B_t = \{b_1, ..., b_u\}$ is a string of bits b_k denoting the presence of a linguistic term l_i^a labeling fuzzy subsets $FS_i^a, i \in [1; K^a]$ in the rule antecedent of the chromosome, where $u, k \in [1; \sum_{j=1}^{g} K^{a_j}]$. ◇

Fuzzy rules are composed of a chromosome serving as the rule antecedent, and of a consequent that is chosen according to the majority class of the training instances covered by the rule antecedent. Example fuzzy rules are presented in Figure 50a.



	s^{amount}	$s^{duration}$	$s^{premium}$	l^{amount}_{low}	l^{amount}_{high}	$l^{duration}_{low}$	$l^{duration}_{high}$	$l^{premium}$	$p1$
Rule from Eq. 10	1	0	1	0	1	0	0	0	full check
Rule from Eq. 12	1	0	1	1	1	0	0	1	no check

(a) Example chromosomes with corresponding consequents for rules from Equations 10, 12



(b) Example perceptron

Figure 50: Example representation of rules for the GA and NF classifiers

GA1. At this step of GA, in each iteration $t \in \mathbb{N}^+$ a chromosome consisting of a fuzzy rule G_t is randomly generated. Next, the rule error $E(G_t)$ is computed by checking its degree of matching between the rule antecedent of instances from the event log subset L_{pdn} and the generated rule antecedent:

$$E(G_t) = \frac{\sum_{a|c_a \neq c_t} w_a \mu^a_{G_t}}{\sum_a w_a \mu^a_{G_t}}, \tag{13}$$

We utilize a boosting approach from [89] which modifies iterative fuzzy rule learning in an incremental fashion. The idea is to repeatedly train a weak classifier on various training data distributions, which shows a considerable improvement in accuracy of the results. The weights of event instances $e_k, 1 \leq k \leq n$ from the current distribution that are correctly classified by the rule G_t are reduced by a factor reflecting the error rate of the generated rule: $w_k(t+1) = w_k[(1 - E(R_t))/E(G_t)]^{\mu^a_{R_t}}$. Misclassified or uncovered examples keep their original weights. Thereby, boosting increases the relative weight of those examples which are "hard to learn". We calculate the chromosome consequent c_t as an output of the boosting classifier considering the vote of the rule G_t, weighted by logarithmic accuracy, on event instances from the event log L_{pdn} by t-norm (cf. [89]):

$$c_t = \underset{c_m}{\text{argmax}} \sum_{G_t|c_t=c_m} \log \frac{1 - E(G_t)}{E(G_t)} \overset{w(t)}{\underset{j=1}{\min}} \mu^a_{l_j} \tag{14}$$

GA2. For building next generation of chromosomes, the rules that have a good fitness are kept. We calculate fitness function as a product of normalized values of such functions as class coverage CC, rule coverage RC, and rule consistency RCS:

$$f(G_t) = \overline{CC(G_t)} \times \overline{RC(G_t)} \times \overline{RCS(G_t)} \tag{15}$$

The class coverage CC is defined as the ratio of the number of training instances covered by the rule G_t to the overall number of training instances carrying the same class label c_i: $CC(G_t) = \sum_{a|c_a=c_t} w_a \mu^a_{G_t} / \sum_{a|c_a=c_t} w_a$.

$$RC(G_t) = \begin{cases} 1, & n > k_{cov}; \\ \dfrac{1}{k_{cov}} \dfrac{\sum_{a|c_a=c_t} w_a \mu^a_{G_t}}{\sum_a w_a}, & o/w. \end{cases} \tag{16}$$

The rule coverage RC reflects the fraction of instances covered by the rule k_{cov}, irrespective of the class label (see Equation 16). If $M \in \mathbb{N}^+$ is a number of classes having the same number of instances in the event log, a reasonable choice for fraction of covered instances is $k_{cov} = 1/M$, as no rule can cover more than such fraction of instances without covering other (false) instances. For example, the number of classes for decision p_1 (see Figure 47) is equal to 3, therefore, k_{cov} is chosen as 0.33.

$$RCS(G_t) = \begin{cases} 0, & n_c^+ \times \varepsilon < n_c^-; \\ \dfrac{n_c^+ - \dfrac{n_c^-}{\varepsilon}}{n_c^+}, & o/w. \end{cases} \tag{17}$$

The rule consistency RCS demands that a rule covers a large number of correctly classified weighted instances $n_c^+ = \sum_{a|c_a = c_t} w_a \mu^a_{G_t}$, and a small number of incorrectly classified weighted instances $n_c^- = \sum_{a|c_a \neq c_t} w_a \mu^a_{G_t}$. Herewith, parameter $\varepsilon \in [0;1]$ determines the maximal tolerance for the error made by an individual rule (see Equation 17). Specific for our use case, as it contains not so many decision classes, we choose ε=0.2. If classes from the log have unevenly distributed number of instances, it is advisable to choose smaller values of ε.

GA3. The GA algorithm iterates t=t+1 and repeats steps *GA1*, *GA2* until a given condition is fulfilled. The termination conditions can be chosen by a process analyst, depending on the requirements stemming from the business environment. For example, the algorithm can stop if a prespecified minimal rule coverage is reached. Also, if the number of rules, that are *not* added to the rule set, reaches a prespecified threshold, the algorithm can be stopped in order to avoid obtaining only redundant or low-performing rules. We discuss in the evaluation section, on the example of our use case, the possibility of combining termination criteria related both to properties of rule sets, and resulting fuzzy decision tables into which the discovered rules are transformed later.

7.3.4.2 *Neurofuzzy Algorithm NEFCLASS (NF)*

Neural networks are also broadly applied to solve fuzzy classification problems, and the NEFCLASS algorithm (NF) is one of the most widely used [89]. NF is also known for producing simple and comprehensible fuzzy rules [133], which is appropriate for the business environment. Below we consider the adaptation of NF applicable to our problem.

NF0. For using NF, we firstly need to represent our problem in a neuro-fuzzy terminology. In such a way, we view the system assisting with mining fuzzy decision rules as a *3-layer fuzzy perceptron*, or simply *perceptron*. Again, given is a process decision (A_{pdn}, C_{pdn}) where

$A_{pdn} = \{A_1, ..., A_g\} \subseteq A$, $1 \leq g \leq v$ is a set of influencing attributes from the corresponding event log subset L_{pdn}. An example perceptron is visualized in Figure 50b.

Definition 27 (Perceptron). A *perceptron* is a network representation of fuzzy classification problem in the form of a neural network $\Pi = (U, \Omega, \Upsilon)$, where

- $U = U_1 \cup U_2 \cup U_3$ is a union consisting of a set $U_1 = \{A_1, ..., A_g\}$ of attributes influencing the process decision (input neurons), a set $U_2 = \{B_1, ..., B_r\}, r \in \mathbb{N}^+$ of fuzzy rules (hidden neurons), and a set $U_3 = \{c_1, ..., c_z\}, z \in \mathbb{N}^+$ of the process decision classes (output neurons);

- $\Omega(a, b) = \mu^a_b$ is a fuzzy weight defined as the membership grade of the value of input neuron $a \in U_1$ in a fuzzy subset provided by the hidden rule unit $b \in U_2$;

- Υ is a mapping that assigns activation functions as follows: $\Upsilon_b = \min_{a \in U_1} \Omega(a, b)$ if $u \in U_1 \cup U_2$, and $\Upsilon_c = \dfrac{\sum_{b \in U_2} \Omega(B, c) \Upsilon_b}{\sum_{b \in U_2} \Omega(B, c)}$ if $u \in U_3$. $\diamond$

NF1. At each step of the algorithm, an event instance from the event log subset L_{pdn} is chosen consequently. Based on this instance, a rule $B_t : A_1 \; is \; l^{A_1}_{x_1}, \ldots, A_{pdn} \; is \; l^{A_{pdn}}_{x_{pdn}} \longrightarrow c_t$ is generated, where A_{pdn} is a subset of influencing attributes, and c_t is the label of the process decision (A_{pdn}, C_{pdn}), that is the output of the rule B_t, $t \in \mathbb{N}^+$,. Further, initialization of the perceptron should be done by creating g nodes in input layer $U_1 = \{A_1, ..., A_{pdn}\}$ corresponding to each attribute in the generated rule. Each input neuron $a \in U_1$ is characterized by K^a fuzzy sets FS^a_i, $i \in [1; K^a]$. Thus, at the iteration t, for each input neuron $a_j \in U_1$ the membership function is found such that $\mu^a_{l_j}(a) = max_{i \in \{1,...,K^a\}}\{\mu^a_i(a_j)\}$. If the hidden layer U_2 of the perceptron does not have a hidden rule node $b \in B$ such that $\Upsilon(A_1, b) = \mu^{a_1}_{l_1}, ..., \Upsilon(a_g, b) = \mu^{a_g}_{l_g}$, then such node is created. Hereby, the class c_t is assigned as the consequent of the rule.

NF2. When adding the generated rules to the outcome rule base, only the rules that have good fitness are kept. For tuning the perceptron weights, we apply the rule B_t on each instance $e_w = (A^w, c^w), w \in \mathbb{N}^+$ from the event log L_{pdn} with the overlapping antecedent and compare the factual class assigned from the event log c^w, and class predicted by the perceptron. We identify the rule fitness as the ratio of the number of the correctly classified instances by the rule $TP(B_t)$ to the sum of number of the correctly and incorrectly classified instances $FP(B_t)$ in the event log subset L_{pdn}:

$$f(B_t) = \frac{TP(B_t)}{TP(B_t) + FP(B_t)} \tag{18}$$

For example, in our further implementation we only keep rules with fitness $f(B_t) \geq 0.1$.

NF3. The algorithm iterates $t{=}t{+}1$ and repeats steps *NF1*, *NF2* until a given condition happens. Again, different termination conditions can be chosen with respect to the business environment. As a new rule is generated for each event instance, the algorithm can be stopped when the complete event log is processed. As this might produce a large amount of rules, the number of first rules to be generated can be prespecified. More complex termination criteria for the NF classifier can be found, e.g., in [89].

7.3.5 *Transformation of Fuzzy Rules into FDT rows*

The outcome of application of fuzzy learners described at the previous step are fuzzy rule sets corresponding to each discovered process decision (A_{pdn}, C_{pdn}), $1 \leq pdn \leq PDN$, therewith, PDN is the number of discovered process decisions, and $A_{pdn} = \{A_1, ..., A_g\} \subseteq A$, $1 \leq g \leq v$ are attributes influencing them. For each of the process decisions (A_{pdn}, C_{pdn}), a corresponding set of frn fuzzy rules is discovered: $R_{pdn} = \{R_1, ..., R_{frn}\}, frn \in \mathbb{N}^+$. Next, the discovered fuzzy rule sets need to be transformed into fuzzy decision tables (FDTs), which are to be used further during process execution. Therefore, the FDT *interpretability* is of the highest importance. However, direct application of discovered fuzzy rule sets might provide low FDT interpretability because of duplications and overlapping of rules and attributes. Below, we describe these issues in more detail and propose the ways to overcome them. Some of the steps are similar to the stages of manual designing of fuzzy decision tables described in [193].

Step 1: Removal of duplications. Duplicated rules lead to FDT with duplicated rows. Therefore, at the first step we remove all duplicate rules.

Step 2: Splitting CNF Rules. Some fuzzy rule learners, as the genetic one, might generate the rule bases consisting of fuzzy rules represented by CNF. As the CNF rules can represent very complex structures, to improve the linguistic interpretability, we propose to replace them by a set of equivalent elemental rules. In particular, each discovered rule $R_y^{pdn} \subseteq R^{pdn}, 1 \leq y \leq frn, y \in \mathbb{N}^+$ can be represented in the following form: $R_y^{pdn} = (\bigcup_{j,i} (A_j, \mu_{l_i}^{A_j}(A_j), \mu_{l_i}^{A_j}), c_{y,z})$, $1 \leq j \leq g, 1 \leq i \leq K^{A_j}$. Then, the corresponding set of elemental rules RS_y^{pdn} can be identified as the Cartesian product of (1) all possible subsets of the rule antecedent containing all attributes of the rule in a couple with the single value of the corresponding fuzzy subset; and (2) the value of the rule class:

$$RS_y^{pdn} = \bigcup_{j,i}(A_j, \mu_{l_i}^{A_j}(A_j)) \times c_y,\ 1{\leq}y{\leq}frn,\ y{\in}\mathbb{N}^+ \tag{19}$$

For example, let the following rule R be given in natural language:

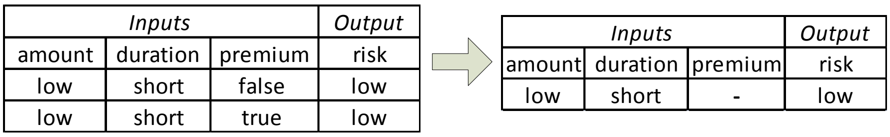
$$R\text{: } amount\ is\ high\ or\ low,\ duration\ is\ short\ or\ long \longrightarrow \\ risk = high$$

Then, the corresponding set of simple rules is $RS = \bigcup_i R_i, 1{\leq}i{\leq}4$,

$$R_1 : amount\ is\ high,\ duration\ is\ short \longrightarrow risk = high$$
$$R_2 : amount\ is\ low,\ duration\ is\ short \longrightarrow risk = high$$
$$R_3 : amount\ is\ high,\ duration\ is\ long \longrightarrow risk = high$$
$$R_4 : amount\ is\ low,\ duration\ is\ long \longrightarrow risk = high$$

Step 3: Mapping simple rules to FDT rows. Further, each elemental rule RS_y^{pdn}, $1{\leq}y{\leq}frn$ is mapped to a corresponding row in a FDT which represents a union of all the fuzzy elemental rules $\bigcup_y\ {RS_y}^{pdn}$. If a rule has no linguistic literal for an attribute, the corresponding cell is left blank, meaning that this attribute has no influence on the outcome.

Step 4: FDT optimization. There can be rules represented by multiple rows, which differ only in the irrelevant attributes' values. If we find such rules, we aggregate them into one row by removing these attributes, because they do not impact the outcome. For example, see Figure 51 where the contracted FDT is derived from the expanded FDT by combining logically adjacent rows that leads to the same action configuration.



Inputs			*Output*
amount	duration	premium	risk
low	short	false	low
low	short	true	low

Inputs			*Output*
amount	duration	premium	risk
low	short	-	low

Figure 51: Example of rule reduction in a FDT

7.3.6 *Identification of Hit Policies for FDT*

Each discovered fuzzy rule, for which its antecedent matches the runtime input, contributes to an output through the compositional rule called *inference*. In DMN, inference is described by hit policies which specify how the table output is obtained, if there are multiple rule matches for a given set of inputs (cf. Definition 13). The activation of a hit policy corresponds to the phase of process execution. During instantiation of processes, the process activity that invokes a corresponding decision model supplies the decision-making system with input data which can be crisp or fuzzy. Below we propose a hit policy formula for FDT.

Given is a process decision (A_{pdn}, C_{pdn}), $1 \leq pdn \leq PDN$, where PDN is a number of discovered process decisions, and $A_{pdn} = \{A_1,...,A_g\} \subseteq A$, $1 \leq g \leq v$ are attributes influencing them. For the process decision (A_{pdn}, C_{pdn}), a corresponding FDT consisting of a set of discovered fuzzy decision rules $R^{pdn} = \bigcup_j R_j, j \in \mathbb{N}^+$ is processed according to the procedures from the previous section. Let an event instance e occur further during the process execution. Then, the activation $\mathcal{A}$ for a rule $R_j, j \in \mathbb{N}^+$ describes the probability of a value from the class C_{pdn} to be correct for the given instance e. For calculating the activation rule value, we propose to utilize an adapted "min-max" operator, one of the most widely used composition operators suitable for Mamdani rule bases [209]:

$$M(R_j) = \min_k [\bigcup_{k \in [1;g]} \max_i (\bigcup_{i \in [1;K^{a_k}]} \mu_{l_i}^{a_k})], \tag{20}$$

where $1 \leq k \leq g : a_k \in e,\ a_k \in A_{pdn}$, so only attributes of an event instance which influence the process decision are evaluated.

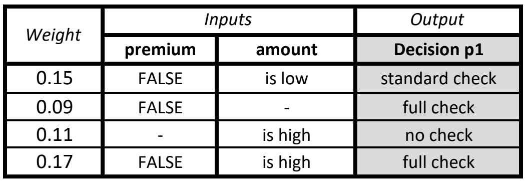
For example, let the loan application process be executed with the following instance data: $amount = 200[EUR]$, $duration = 30[mths]$, $premium = TRUE$. Next, we calculate the value of the activation rule from Equation 11. Here k=2, and, consulting the membership functions (see Figure 47), we establish that $\mu_{low}^{amount} = 1$, $\mu_{high}^{amount} = 0$, and $\mu_{short}^{duration} = 0.6$, $\mu_{long}^{duration} = 0.4$. Then, the value of the activation rule from Equation 11 is calculated as follows: $M(R) = \min\ [\bigcup \max(\{1;0\},\{0.6;0.4\})] = 0.6$. Analogously, the activation values for all the rules in the set $R^{pdn} = \bigcup_j R_j, j \in \mathbb{N}^+$ are evaluated, and the rule maximizing the value of the activation rule is chosen.

Let several rules overlap in the discovered fuzzy decision table, and let the activation rule from Formula 20 return the same value for the overlapping rules. In such case, it is possible to use the *collect* hit policy (cf. Section 2.3). Then, the output result is the list of all the matching output values.

However, if an unambiguous single output is desired, we propose to additionally weight the discovered rules by their error rate on instances from the event log, according to the following formula:

$$\mathcal{A}(R_j) = W(R_j, L_{pdn}) * M(R_j) \tag{21}$$

In the formula above, the weight variable $W(R_j, L_{pdn})$ characterizes how good the rule classifies the instances from the event log subset L_{pdn} corresponding to the process decision (A_{pdn}, C_{pdn}). The calculation of the weight value depends on the applied fuzzy learner. For GA, this value is equal to the logarithmic accuracy of the rule $\log(1 - E(R_j))/E(R_j)$ from Equation 13. For the NF, this value is equal to the rule fitness $f(R_j) = TP(R_j)/(TP(R_j) + FP(R_j))$ from Equation 18. Assigning the weights to rules in such a way can also contribute to satisfying the condition of *exclusivity* of decision rules in FDT, if that is required by user. Then, the rows are sorted by their weights, and while iterating over the rows, a rule is removed if it overlaps with the previous one. The low quality rows can also be removed, if some threshold is established.



Weight	*Inputs*		*Output*
	premium	**amount**	**Decision p1**
0.15	FALSE	is low	standard check
0.09	FALSE	-	full check
0.11	-	is high	no check
0.17	FALSE	is high	full check

Figure 52: Example of a fuzzy decision table that consists of weighted rules, which can be discovered from an input event log. The fuzzification of the matching input values for the numeric inputs is executed during run-time, with the help of the predefined membership function (see Figure 47). If multiple outputs are allowed, then the hit policy can be of the customized collect type, which incorporates the fuzzy activation operator (see Formula 20). However, in the cases when only one output is required, the hit policy is provided by Formula 21, which incorporates the fuzzy rule activation operator weighted on the error rate of the discovered rules in the event log.

For an example of applying the fuzzy hit policy proposed by us in Formula 21, consider the decision table from Figure 52. This fuzzy decision table is an example table that can be derived from the event log of the loan application process. The weights of the decision rules, represented in the first column of the table, are example weights that can be derived from the event log in accordance to the formulas from the previous paragraph. Further, let the loan application process be executed with the instance data $premium = FALSE$, and $amount = 200[EUR]$.

As mentioned in Section 7.3.2, the membership functions of the nominal attributes are represented by their characteristic function. Therefore, for the *premium* variable from our example, we establish that $\mu^{premium}_{FALSE} = 1$. Consulting the membership function for the *amount* variable (see Figure 47), we establish that for the value $amount = 200[EUR]$, the membership function values are: $\mu^{amount}_{low} = 1$, $\mu^{amount}_{high} = 0$. The first case corresponds to a higher value of the membership function. Therefore, the value $amount = 200[EUR]$ is mapped to the expression *amount is low*.

Further, we observe that the rule predicate $premium = FALSE, amount\ is\ low$ matches the first two rows of the table from Figure 52. Therefore, we detect a case of the overlapping rules matching the same input. The usage of the weights provides us with an additional information, which allows us to make a choice between these rules.

Non-weighted activation rules of the two matching fuzzy rules can be calculated with the help of Formula 20, that is $M(R_1) = \min\ [\bigcup \max(\{1\},\{1;0\})] = 1$ and $M(R_2) = \min\ [\bigcup \max(\{1\},\{1;0\})] = 1$. Since the non-weighted activation rule for fuzzy rule inference returns the same result, the overlapping is not resolved. As mentioned above, if the collect hit policy is assigned to this table, then a set of matching outputs is returned. However, in this example it is reasonable to require that only one output should be chosen, as the types of checks in the loan application process are exclusive. Therefore, we apply the weighted activation rule from Formula 21 using the weights of the rules from Table 52, that is $\mathcal{A}(R1) = 0.15 * 1 = 0.15$, and $\mathcal{A}(R2) = 0.09 * 1 = 0.09$. Since the value of the weighted activation rule for the first rule is higher, this rule should be applied, which yields *standard check* as the decision output.

With respect to the DMN standard, all the DMN hit policies can be applied to FDMN. Fuzzy activation rules are not foreseen yet. However, the standard recommends to implement custom post processing steps in combination with the *collect* policy, which can be used considering the activation rule from Equation 21.

7.4 APPLICATION OF THE APPROACH ON THE EXAMPLE LOG

To validate our methodology to discover fuzzy decision models from an event log presented in Section 7.3, we implemented it and applied it to a simulated event log. The details of our implementation, and the evaluation experiments can be found in Chapter 8. In the following section, we discuss application of our approach to the motivating example presented in Section 7.1.

Event Log Simulation

For analyzing the example business process, we again simulated the event log with the help of the CPN Tools software. The newly generated event log contains two numeric attributes *duration* and *amount*, and two nominal attributes *premium* and *risk*.

To be able to create an event log with the simulation parameters for some of the process decisions described in linguistic terms, we assign linguistic terms to the input numerical attributes and define the corresponding membership functions. This is an essential step, as we

Linguistic literal	x_1	x_2	x_3	x_4
low	MIN_INT	MIN_INT	20	50
high	20	50	MAX_INT	MAX_INT

Table 11: Values for defining points of the trapezoidal membership function for the numeric attribute *amount* (the units used are EUR)

Linguistic literal	x_1	x_2	x_3	x_4
short	MIN_INT	MIN_INT	5	10
normal	5	10	10	15
long	10	15	15	20
very long	15	20	MAX_INT	MAX_INT

Table 12: Values for defining points of the trapezoidal membership function for the numeric attribute *duration* (the units used are months)

need to describe the simulation parameters for some of the process decisions in linguistic terms, in order to validate that our algorithms for fuzzy rules discovery, presented in Section 7.3.4, produce appropriate classification results.

Thereby, we need to assign the linguistic terms to them and define the corresponding membership functions. As discussed in Section 7.2, we utilize the trapezoidal membership functions for the demonstration of our approach. The curve for this function is defined by four parameters x_1, x_2, x_3, x_4 (cf. Definition 24). Table 11 presents the parameters of the membership function determined by us for the attribute *amount*. Table 12 presents the parameters of the membership function identified by us for the attribute *duration*. The visualization of corresponding curves are presented in Figure 47.

To create the input event log, we used the distributions of the numerical attributes that were used also in Section 6.5. For the nominal attributes we created corresponding sets of fuzzy rules. The summary of all the simulation parameters is presented in Table 13.

Discovery of FDMN model

As discussed in Section 7.3.3, discovery of DRD models from event logs represents a classification problem over crisp data. We applied the corresponding algorithms from Chapter 6, and discovered the DRD model for the generated log as presented in Figure 53a. Thereby, three process decisions were discovered: (1) control flow decision p_1 with influencing attributes *amount* and *premium*; (2) control flow decision p_3 with the influencing attribute *risk*; and (3) data decision *risk* with influencing attributes *duration* and *amount*.

Task / Attribute Name	Simulation Parameters
Trace ID	1 to 200 (incrementing)
amount	discrete(2,99)
premium	random boolean
duration	discrete(2,30)

risk	if *duration* is *very long* and *amount* is *high* : *very high*
	if *duration* is *long* and *amount* is *high* : *very high*
	if *duration* is *normal* and *amount* is *high* : *very high*
	if *duration* is *short* and *amount* is *high* : *normal*
	if *duration* is *very long* and *amount* is *low* : *high*
	if *duration* is *long* and *amount* is *low* : *low*
	if *duration* is *normal* and *amount* is *low* : *low*
	if *duration* is *short* and *amount* is *low* : *low*

*p*1	if *amount* is *high* and *premium* is *false*: *full check*
	if *amount* is *high* and *premium* is *false*: *standard check*
	if *premium* is *true*: *no check*

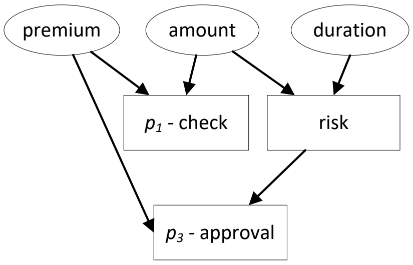
*p*3	if *risk* is *low*: *accept*
	if *risk* is *normal*: *accept*
	if *risk* is *high*: *reject*
	if *risk* is *veryhigh*: *reject*

Table 13: Simulation parameters for generating the event log of the process from Figure 39

Next, we applied both the GA and the NF classifiers for discovering FDTs corresponding to the DRD model from Figure 53a. Thereby, we consequently applied the techniques from Section 7.3.5 for constructing the fuzzy decision tables from the discovered rule bases. An example of a fuzzy decision table for the rule-based data decision *risk* discovered with the help of the GA algorithm is presented in Figure 53b.

The decision rules in the constructed decision tables are assigned with weights that show how good the rule classifies the instances from the event log. In the running example, we utilize this information through the usage of the weighted activation rule as the hit policy for the discovered fuzzy decision table (see Formula 21). For example, during the execution of the process, the numeric input can be mapped, with the help of the corresponding membership functions, to such values of the *duration* attribute as *very long*, and of the *amount* variable as *high*. In this case, the second and the fifth decision rules from Figure 53b can be applied. In this case, the rule output for which the value of the weighted activation rule from Formula 21 is higher, is chosen, and the corresponding output, *risk is very high* or *risk is high*, is returned. An example of the application of Formula 21 is provided in Section 7.3.6.

The decision rules in the discovered fuzzy decision table from Figure 53b classify the event log characterized by the simulation parameters from Table 13 quite well, since 6 out of 8 rules for the *risk* attribute are determined correctly. However, some differences can also be observed. We discuss in details the accuracy of the output FDMN



	Inputs		*Output*
Weight	**duration**	**amount**	**risk**
0.22	is long	is high	very high
0.22	is very long	is high	very high
0.22	is short	is high	normal
0.11	is very long	is low	high
0.26	is very long	is high	high
0.26	is normal	is high	high
0.33	is short	is low	low
0.18	is normal	is low	low

(a) The discovered decision requirements diagram for the example process

(b) Discovered fuzzy decision table for the rule-based data decision *risk*

Figure 53: The discovered decisions and the DMN model for the example process

model by doing a set of evaluation experiments on the simulated event log in Chapter 8.

7.5 SUMMARY AND DISCUSSION

In this chapter, we introduced a novel methodology to mine fuzzy decision models that can be used complementary to process models from event logs. Firstly, we introduced a formal framework describing incorporation of fuzziness in a decision model in Section 7.1. Section 7.3 introduces the steps of deriving a fuzzy DMN decision model from the input event log.

In particular, we presented identification of fuzzy subsets and corresponding membership functions corresponding to the data attributes from the event log. As fuzziness is only relevant for the decision logic of the output decision tables, we discussed that the DRD models represent a classification problem over crisp data. Therefore, the approach to discover DRD models from event logs from Chapter 6 can be reused. Hereby, for learning fuzzy decision rules, we explored application of the genetic and NEFCLASS classifiers. We showed how to construct and simplify fuzzy decision tables. Additionally, we proposed a formula for the fuzzy activation rule as a hit policy for discovered fuzzy decision tables. Usage of FDMN models that consist of crisp DRDs referencing fuzzy decision tables and fuzzy hit policies, as proposed by us, is not yet foreseen by the DMN standard, but is fully compliant with it.

Finally, we validated the proposed methodology on a simulated event log of the example credit-risk assessment process in Section 7.4. The derived FDMN explains in linguistic terms the decision making recorded in the event log. The derived decision model can be automated and executed complementary to the process model. During the execution, process instance data consisting of both numerical and nominal attributes is evaluated against the set of fuzzy decision rules that converts inputs to outputs. Fuzzy decision tables are able to capture vagueness and imprecision of the execution data with the help of membership functions that are mappings describing the uncertainty in the data values by assigning them to predefined linguistic terms with a certain probability.

With regards to the limitations of the proposed methodology, for some steps of it, the domain knowledge is required. For example, an analyst has to describe the fuzzy subsets for the attributes from the event log. Also, it is recommended that an expert tunes the parameters of the fuzzy classifiers, such as termination criteria, or rules fitness threshold, depending on requirements of the business environment. FDMN mining, like other classification problems, often leads to the necessity to find a compromise between interpretability and accuracy of the output model. We conduct a set of experiments to evaluate the interpretability and accuracy of the output decision models resulted from application of our methodology on the example event log later in Chapter 8.

Part III

EVALUATION, DISCUSSION, AND CONCLUSION

8 EVALUATION OF THE DECISION MODEL DISCOVERY METHODOLOGIES

The previous chapters lay the foundations and introduce the methodologies to discover decision models complementary to process models, taking both process models and event logs as inputs. This chapter focuses on the evaluation of the presented methodologies, which constitute the last step of the design science research methodology, applied in this thesis as discussed in Section 1.4. Thereby, we conduct a series of experiments in order to observe how the presented methodologies perform in real-world settings. The chapter is based on results published in [18, 20, 21, 23, 24, 25].

Since the methodologies presented in Chapters 4–7 assume different inputs to the decision model discovery algorithms, we present a dedicated section describing relevant experiments on real-life data, for each technique corresponding to these chapters. The rest of the chapter is structured as follows. Section 8.1 outlines the chapter in detail and provides the description of the applied evaluation methods. Section 8.2 provides the evaluation of the approaches to discover decision models using a process model as input. Section 8.3 provides the evaluation of the approaches to discover decision models using an event log as input. Section 8.4 summarizes the chapter.

8.1 EVALUATION OUTLINE, METHODS, AND METRICS

Below, we provide the structural outline of our evaluation, the applied evaluation methods, and the utilized evaluation metrics.

OUTLINE In Part II of this thesis (cf. Chapters 4–7) we designed techniques to extract decision models by taking into account two perspectives using both process models and event logs as inputs. Proceeding with the design science steps (cf. [204]), in the current chapter we evaluate the applicability and usefulness of the developed methodologies in a real-life environment.

Figure 54 shows the outline of this chapter. The current section describes the evaluation outline and introduces the applied evaluation methods. Next, two different perspectives to discover decision models are considered. Section 8.2 evaluates our approaches to discover decision models taking a process model as an input. Our methodology of control-flow-based decision model discovery from Chapter 4 is evaluated in Section 8.2.2. The data-centric decision

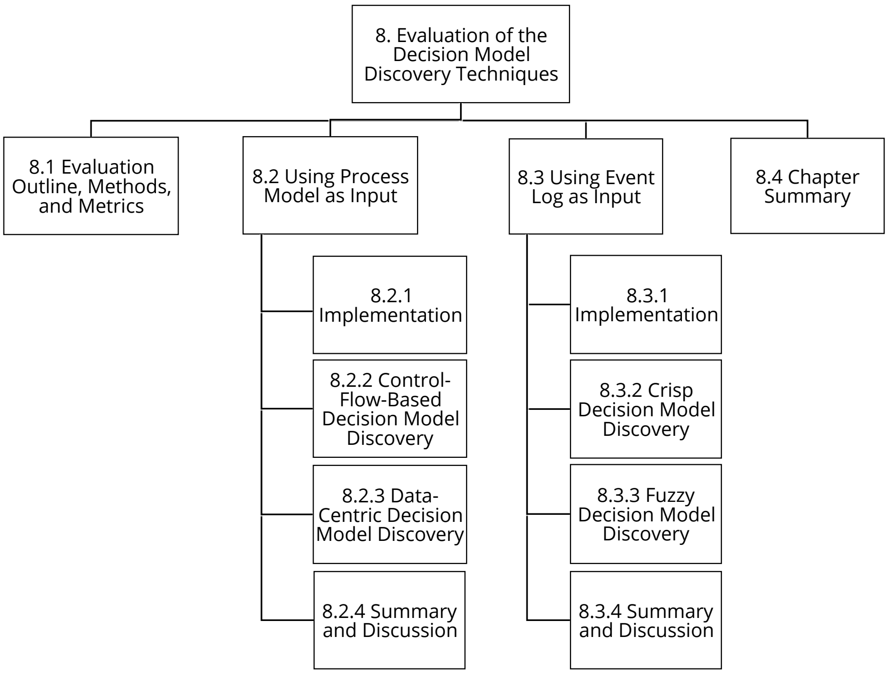

8. Evaluation of the Decision Model Discovery Techniques
8.1 Evaluation Outline, Methods, and Metrics
8.2 Using Process Model as Input
8.3 Using Event Log as Input
8.4 Chapter Summary
8.2.1 Implementation
8.2.2 Control-Flow-Based Decision Model Discovery
8.2.3 Data-Centric Decision Model Discovery
8.2.4 Summary and Discussion
8.3.1 Implementation
8.3.2 Crisp Decision Model Discovery
8.3.3 Fuzzy Decision Model Discovery
8.3.4 Summary and Discussion

Figure 54: Outline of the evaluation of the presented methodologies

model discovery from Chapter 5 is evaluated in Section 8.2.3. In Section 8.3, we evaluate our approaches to discover decision models from event logs. The evaluation of our methodology to discover crisp decision model from Chapter 6 is presented in Section 8.3.2. The methodology to discover decision model extended with fuzziness from Chapter 7 is evaluated in Section 8.3.3.

METHODS To conduct the evaluation experiments, we performed the *design and implementation of the prototypes* validating the presented methodologies. Section 8.2.1 introduces the prototype, which was developed to support the discovery of decision models complementary to process models using a process model as an input. Section 8.3.1 introduces the prototype for the discovery of decision models complementary to process models from event logs.

The evaluation goal is to investigate the *interaction of the developed methodologies with their real-world context*, and thus we considered the application of the presented methodologies by providing, as input into the developed prototypes, the following real-world objects: (1) real-life process model repositories; (2) real-life event logs, and (3) synthetic event logs simulated with the parameters representing a real-life environment.

METRICS The applied evaluation metrics differ by the perspective, from which the decision model is discovered. The methodologies using a process model as input (cf. Section 8.2) are pattern-based. Therefore, the main evaluation metric to determine the usefulness of the proposed methodologies, is the *patterns occurrence frequency* in real-life process model repositories. The patterns occurrence frequency shows if the identified patterns are frequently used in practice [212], and, therefore, if the discovery of decision models based on these patterns has relevance and applicability to real-life context. Additionally, as the control-flow-based discovery incurs a major refactoring of the input process model, the *complexity* of input process models [92] before and after refactoring is evaluated. We measure the complexity with regards to the number of the process model elements, e.g., the number of the split XOR gateways.

The methodologies that discover decision models using event logs as inputs (cf. Section 8.3) are based on solving the classification problem where the classes are process decisions that can be made, and the training examples are the process instances recorded in the event log. One of the main evaluation metrics showing how a discovered model performs is *accuracy* [205]. In particular, in our setting, the decision model accuracy is the average number of event instances that are correctly classified by either decision tables, or functions corresponding to the discovered decisions which constitute the output DRD model, divided by the total number of all instances in the test log. Furthermore, since achieving a better *interpretability* for the output decision model [27] is our motivation to incorporate fuzziness into decision model discovery, we use it as an additional evaluation metric. For evaluating interpretability of the discovered decision models, we take into account such parametres as the number of discovered decision rules, and the presence of dependencies between the discovered decision rules in the output decision model.

8.2 USING PROCESS MODEL AS AN INPUT

In this section, we evaluate the methodologies to discover decision models using process models as input (cf. Chapters 4–5). The results presented in this chapter are based on a collaborative project with our partners who provided us with the real-world process model repositories, extracted from the various industries. Section 8.2.1 presents the implementation details of the research prototype to discover decision models using process model as an input. The implementation is co-authored by all contributors from [18]. Further, we utilize the presented prototype for conducting a series of the evaluation experiments, which is a novel contribution of this thesis. The evaluation of the control-flow-based discovery of decision models from process models is presented in Section 8.2.2. The experiments pertaining to the data-centric discovery of decision models are presented in Section 8.2.3. Section 8.2.4 provides a summary and a discussion of these experiments.

8.2.1 *Implementation*

Our algorithm to discover decision models process models is implemented in the Promnicat [1] framework, an open-source platform for analyzing process model repositories. The framework is written in the Java programming language [11]. This platform provides importing functionality for a set of process collections, modeled by notations widely used in research and industry, e.g., the BPMN standard. The Promnicat framework provides the functionality to explore, transform, and extract information from process model collections [65]. We further extended the functionality of Promnicat by implementing our own *Decision Model Discovery* plug-in, for the discovery of decision models from process models. The implementation, associated documentation, and several example process models are available at *http://bpt.hpi.uni-potsdam.de/Public/BpmnDmn*.

The architectural overview of our implemented plug-in, to discover decision models, is presented in Figure 55 with the help of a FMC *block diagram*[1]. Block diagrams represent active system components called agents (rectangular shape) and passive system components called locations (rounded shape). Agents communicate with each other via channels depicted as lines with a small circle. The combination of a solid triangle and the letter *R* next to the circle indicates that one agent request information from another agent which in turn responds (the triangle vertex points to the receiver).

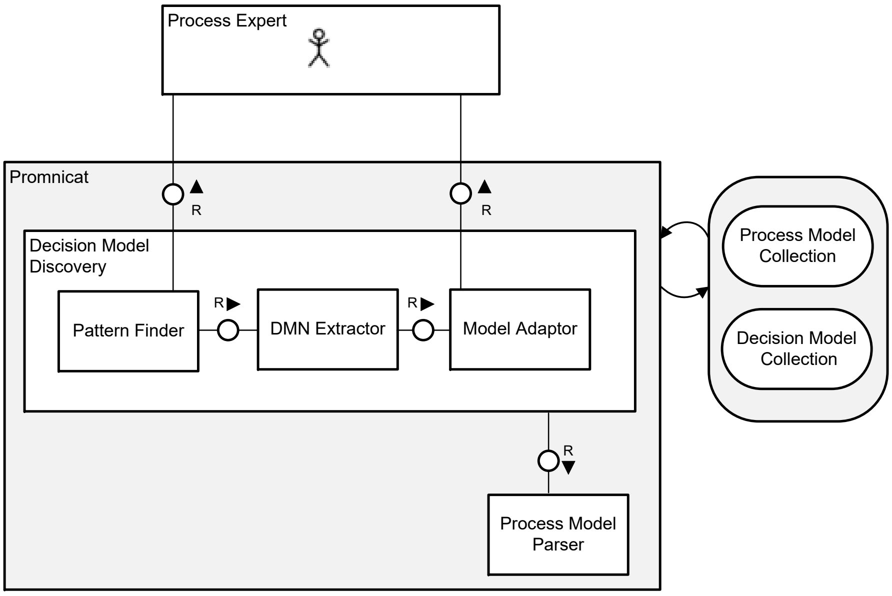


Figure 55: FMC block diagram depicting the implemented Decision Model Discovery Promnicat plug-in

[1] Fundamental Modeling Concepts (FMC) is a modeling notation where block diagrams are used to illustrate compositional structures as a composition of collaborating system components [103]

As can be seen from Figure 55, the *Decision Model Discovery* plug-in extension of Promnicat consists of three blocks: the *Pattern Finder*, the *DMN Extractor*, and the *Model Adaptor*.

The work of the developed prototype starts when a process model is imported from the *Process Model Collection* storage. The process model has to be represented in XML format. Next, the process model is parsed with the *BPMN Parser* component of Promnicat. Further, the *Pattern Finder* component analyses the imported process models and identifies a predefined set of process patterns (e.g., in accordance to the steps from Section 4.4). Additionally, the *DMN Extractor* component obtains DMN decision models (see Section 4.5) which are stored in the *Decision Model Collection* storage. The *Model Adaptor* realizes the post-processing of both process and decision models (e.g., in accordance to Section 4.6). The process expert is then used for choosing one of the overlapping patterns discovered by *Pattern Finder*, and performing the linguistic adaptation of process models that are refactored by the *Model Adaptor*. The output decision models are also stored in XML format. Promnicat does not have a built-in visualizer, but the output process and decision models can be visualized by any suitable external process model editor supporting the import of XML models, e.g., Signavio Process Explorer [172].

We used this implementation to evaluate the impact of our approaches to discover decision models from the real-life process model repositories, which are presented in the following sections.

8.2.2 *Control-Flow-Based Decision Model Discovery*

In this section, we evaluate the methodology to discover decision models from the control flow of process models presented in Chapter 4. To evaluate the presented methodology, we conducted a series of experiments on the real-life process model repositories obtained from our project partners, which were introduced in Section 4.3.1.

Firstly, for every process model in the each of the repositories, the control-flow-based pattern identification from Section 4.4 was applied. During this step, all the detected patterns were considered, even if they were overlapping each other. The relative frequency of detected control-flow-based patterns (cf. Section 4.3) was calculated, as presented in Figure 56. The detailed numerical results of this evaluation experiment can be found in Table 15 from Appendix B.

It can be seen from Figure 56 that pattern $P1$ is the most frequent pattern. In average, it was identified in 63.26% of the process models of the real-life process model repositories. Pattern $P2$ was detected in 17.83% of the analyzed process models. Finally, pattern $P3$ was identified in 10.04% of the considered process models. These numbers show that the identified patterns are frequently used in practice, and

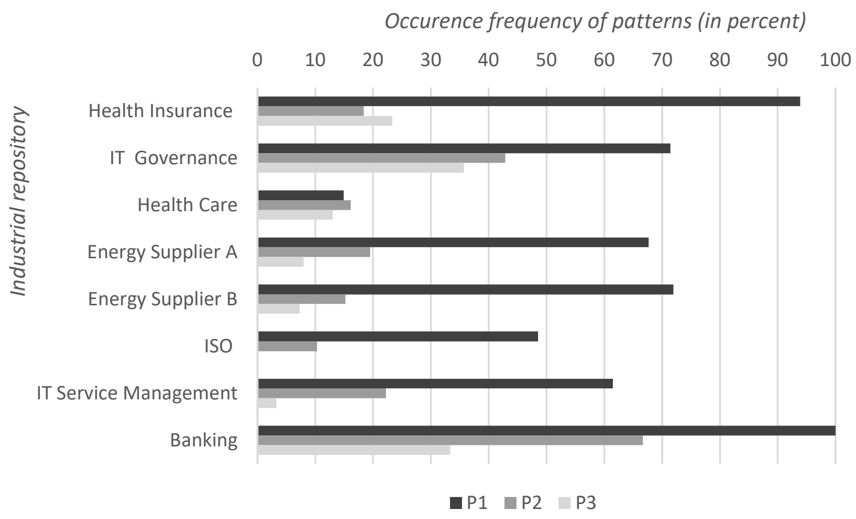


Figure 56: Relative frequency of control-flow-based patterns from the real-life process model repositories presented in Section 4.3.1

thus, the discovery of decision models based on these patterns has relevance and applicability to real-life context.

Next, we conducted the steps of decision model extraction from the set of identified patterns (cf. Section 4.5) and post-processing of process and decision models (cf. Section 4.6). For the evaluation of this part of the introduced methodology, we measured the rate of the refactored models in the given real-life process model repositories. We define this rate as the number of models for which both process model refactoring and decision model creation took place. The results are presented in Figure 57. The detailed numerical results of this evaluation experiment can be found in Table 16 (see Appendix B).

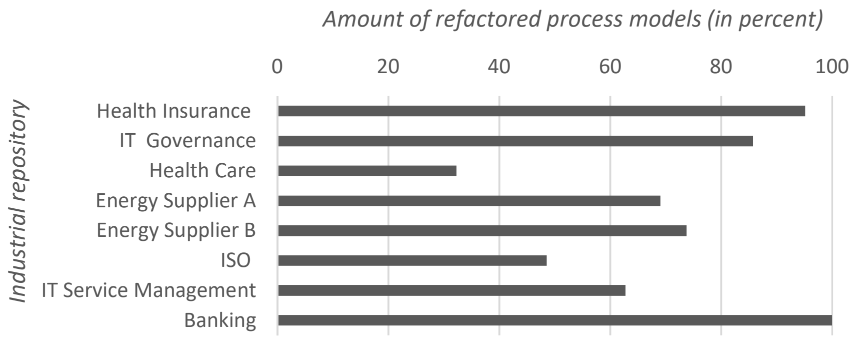


Figure 57: Relative percentage of refactored models in the real-life process model repositories from Section 4.3.1

The weighted average rate of the refactored models in the considered repositories is 67.12%. Thus, more than two-thirds of the considered process models were modified, and the decision models were generated successfully.

To get more insights into the effect of the performed model transformations, we calculated the occurrence ot the split XOR gateways in the process models before and after refactoring of the process models in the real-life repositories. Figure 58 visualizes the results, whereas the detailed numerical results of this evaluation experiment can be found in Table 17 from Appendix B.

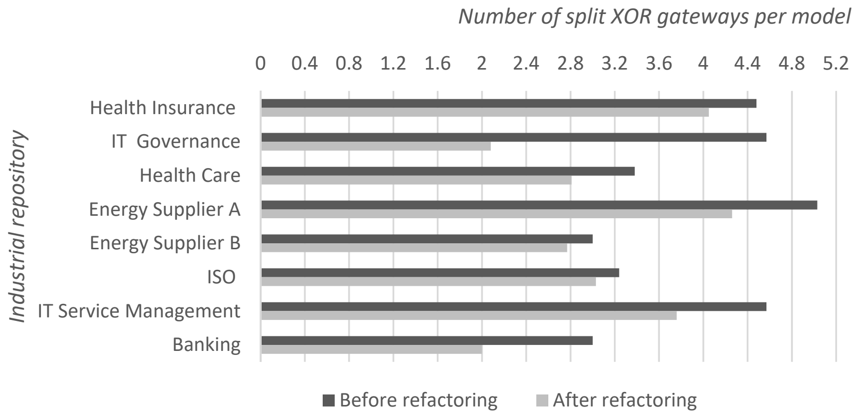


Figure 58: Number of the split XOR gateways per model in the real-life process model repositories from Section 4.3.1 before and after the refactoring experiment

On average, the number of split XOR gateways reduced by 20.81% in the process models after their refactoring. This result is achieved by externalizing the complex XOR structures into dedicated decision models. This reduction rate implies a reduced number of process model elements, thereby significantly improving the readability of the refactored process models.

We can therefore conclude, that the application of our methodology for extracting decision models from the control-flow of process models increases the readability and flexibility of real-life process models. Also, the extracted decision models provide a more structured design for process decision making, and they can be executed complementary to process models which realizes the separation of concerns principle.

There are, nevertheless, also limitations of the proposed methodology with respect to its application in a real-world environment. Firstly, as discussed by Assumption 2 of Section 4.2, we assumed that decisions are taken in distinct process model activities preceding the split XOR gateways in the control-flow-based patterns. However, our manual consideration of the process model repositories revealed that sometimes the decision activities were omitted in the process models, and we had to insert them manually. As a result, depicted in Table 17 from Appendix B, the average number of activities in the refactored process models has increased by 2.14%. In such a way, manual checking of the real-life repositories for their correctness is required. Additionally, as discussed in Section 4.4, involvement of an expert is required for resolving the conflicts which occur if several control-flow-based patterns match overlapping process model fragments.

Another limitation originating from the design of our methodology is connected to the post-processing of process and decision models after extraction of the decision model. As discussed in detail in Section 4.6, the linguistic adaptation of process and decision labels after process model refactoring is needed, which at the current moment is supposed to be done by a process expert.

8.2.3 *Data-Centric Decision Model Discovery*

In this section, we evaluate the methodology to discover decision models from the data flow of process models presented in Chapter 5. Many real-life process models are activity-centric, and further modeling artifacts, such as data, are often not supported [128]. The process model repositories from the previous experiments were no exception to this. Therefore, to evaluate our data-centric decision model discovery, we considered another repository, referred to as the hospital repository, which was rich in data-centric real-life process models. Thus, our partners provided us with a set of real-life process models, modeled for a project which had the goal of establishing a business process management system in a hospital's surgery department. The project was a collaborative effort of practitioners and process model experts, and therefore, the data output process models were well defined. The details of this project can be found in the publication of Kirchner et al. in [98].

In total, we considered a set of 43 process models. All of them were dedicated to a rather complex liver transplantation process and its supporting processes. All the process models provided to us were designed, with the help of BPMN, by a collaborative team of physicians and process designers. Examples of the corresponding process models can be found in [98]. The process models were provided to us as pictures, therefore, it was not possible to conduct the parsing of the process models and we conducted the manual analysis of the repository.

Firstly, for each process model in the hospital repository, the data-centric decision pattern identification from Section 5.3.3 was applied. Thereby, the relative frequency of detected data-centric decision patterns *Π1*–*Π6* (cf. Section 5.3.2) was calculated, as presented in Figure 59. The detailed numerical results of this evaluation experiment can be found in Table 18 from Appendix B.

As can be seen from the figure, the most highly used patterns are pattern *Π2* which was detected in 44.19% of the process models, and *Π1* which was detected in 39.53% of the process models. Thus,

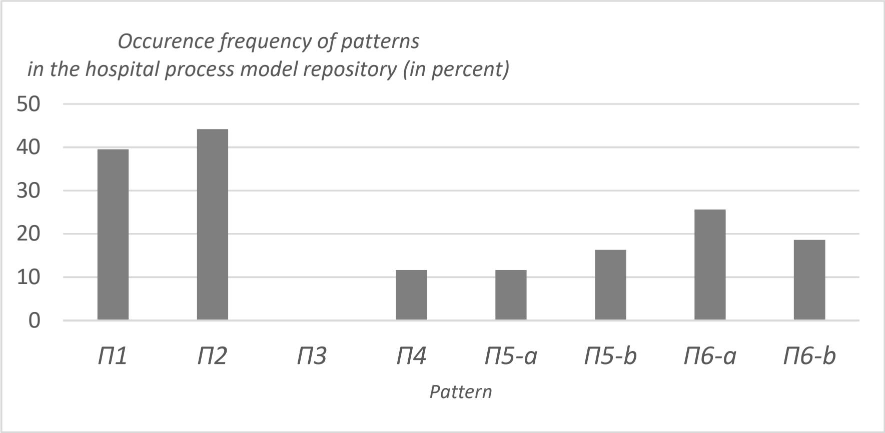


Figure 59: Relative frequency of data-centric decision patterns in the hospital repository

the usage of text annotations (*Π2*) and data nodes (*Π1*) are the most commonly used data structures to incorporate decision making.

The second most frequent group of patterns consists of pattern *Π6−a* which was detected in 25.58% of the process models and pattern *Π6−a* which was detected in 18.60% of the process models. However, our observation shows that these two patterns incorporating boundary events and the corresponding exceptional flows were frequently misused for modeling the non-exceptional control flow. Although this fact does not influence the derivation of decision models according to the methodology proposed by us, the stakeholders that are interested in improving the quality of their models could take this into account and conduct a model re-design.

Patterns *Π5−a*, detected in 11.63% of the process models, and *Π5−b*, detected in 16.28% of the process models, were slightly less common. This can be explained due to the fact that start and intermediate events, preceding the decision activities, most often serve as the triggers of decisions rather than bearing decision-related data.

Pattern *Π4* representing the involvement of a resource in the decision making was detected in 11.63% of the process models. This can be explained by the fact that the hospital does not provide many resources for the execution of one clinical pathway. It has to be noted that the influencing resource was sometimes related to the decision activity through an icon of the responsible person attached to the activity, and not through process lanes.

It can also be seen from Figure 59 that pattern *Π3*, which represents data stores used by a decision activity, was not detected in any of the process models of the considered repository. This can be explained by the fact that the domain experts involved in the process model design were the hospital physicians who did not deal extensively with IT-systems. Thus, several years ago from now, the electronic patient files were not so common in the hospital, and the majority of the documentation was paper-based.

Next, we summarize the evaluation results of data-centric decision model discovery using a process model as an input. On the one hand, embedding of a major part of process decision making by data-centric decision patterns could be explained by the fact the process models were designed under time pressure in a series of workshops [98]. On the other hand, the process designers were not aware of the principle of separation of process and decision concerns, especially due to the fact that the process models were designed several years when the DMN standard did not exist yet. Thereby, in average, the data-centric decision patterns occurred quiet frequently in the considered repository. In such a way, the identified patterns present suitable candidate fragments to be externalized into a dedicated decision model.

Following the evaluation of the patters, we manually conducted the step of decision model extraction from the set of identified patterns (cf. Section 5.4) and post-processing of process and decision models (cf. Section 5.4.2). Again, for the evaluation of these steps, we measured the rate for both process and decision models that were refactored. In total, 32 process and decision models were refactored, which is a refactoring rate of 74.42%. Thus, almost three quarters of the considered process models were refactored, and the decision models were generated successfully.

We can therefore conclude, that the application of our methodology to discover decision models from process model data shows positive results. The extracted decision models describe process decision making in a compact way, and they can be executed further, complementary to process models which realizes the separation of concerns principle.

The presented approach has certain limitations though. Firstly, the process or domain experts have to manually identify which activities of the input process model are decision activities (cf. our assumption from Section 5.2). This limitation is due to the fact that the BPMN standard does not provide a mechanism to explicitly demarcate decision activities, e.g., the user task type activities could also incorporate decision making but do not have to. Secondly, during the refactoring of the process models, we had to add 16 extra decision activities since they were omitted in the original models. This corresponds of a rate of 0.37 decision activity per process model from the repository. This step of our methodology also required an expert's input.

According to our post-processing recommendations from Section 5.4.2, an analyst identifies if the elements of the detected patterns should be kept in the input process models, or if they should be fully externalized as a dedicated decision model. Our analysis of the hospital process model repository shows that the data is not misused for modeling decisions, but rather has an informational function. This is in contrast to the misuse of control flow for modeling the decision logic revealed to us earlier (cf. Chapter 4). Thus, during the first extraction of a decision model from for the input process model, we recommend to keep the decision-related data elements in the process model. Since the process are executed mostly manually, extra recommendations might ease the understanding and usage of the process models by users. Once the staff gets sufficient experience with the process models, and the processes become more automated, we would recommend further simplification of process models by excluding the elements present in the decision models from process models. Future works could address this problem in a more systematic way.

8.2.4 *Summary and Discussion*

The section above considers the perspective of discovering decision models using process models as input. This perspective incorporates both control-flow-based decision model discovery, presented in Chapter 4, and data-centric decision model discovery, presented in Chapter 5.

The first evaluation method that we use is the design and implementation of a prototype validating our methodologies. The Promnicat framework is an open-source platform that is designed specifically for research on process model repositories. Therefore, we extended Promnicat with the *Decision Model Discovery* plug-in which is used for our experiments.

The second evaluation method that we use is the investigation of how our methodologies interact with the real-world context. In order to do this, we conducted a series of experiments on a set of available real-life process model repositories from several industries.

Since the methodologies to discover decision models using process model as an input are pattern-based, we chose the *patterns occurrence frequency* as the main evaluation metric. The real-life repositories reviewed by us consist in total of 1159 process models. The developed control-flow-based patterns are detected in the repositories in 10% to 63% of all the process models. The pattern representing a single split gateway is the most commonly occurring pattern, which is detected in more than 63% of all the process models.

The data-centric patterns occur in the considered hospital process repository in 11% to 44% of the process models. From this range, we excluded the occurrence frequency for the pattern representing a data store used by a decision activity which was not detected in the repository. This is possibly due to the fact that the domain experts involved in the process model design did not deal with the hospital IT-system. The most frequently used data-centric pattern to describe decision logic is by using a text annotation which was detected in more than 44% of all the process models in the repository.

Overall, the patterns occurrence in the considered repositories demonstrate that the identified patterns are frequently used in real-world. In all the cases, there is possible explanation for the extensive encoding of the decision logic, either in the control flow or the data of process models. It is due to the fact that the domain experts involved in the process model design were not provided with the instruments to separate decision logic from process logic. Thus, the DMN standard appeared after the considered process model repositories were created. In such a way, the usage of the considered patterns as a base to discover decision models complementary to process models is valid and provides the potential for application in practice.

After detecting the decision patterns in the process models, we were able to refactor more than two thirds of all the considered process models by externalizing the decision logic into dedicated decision models, which can be used complementary to process models. Thereby, it was confirmed that, with the help of the methodologies developed by us, it is feasible to bring into practice the principle of separation of process and decision concerns (cf. Section 2.3.3). The advantage of applying this principle can be especially well seen for the control-flow-based decision discovery. Since the control flow of the considered process models was clearly misused for decision modeling, the corresponding control flow structures were externalized into dedicated decision models. This provide a significant simplification of the input process models. Thus, the number of the split XOR gateways was reduced for more than 20% of the considered process models. A broader implication of reducing the complexity of process model to such an extent means that the provided methodologies can serve as tools for business process improvement, the effect of which could be explored in future works.

The evaluation experiments introduced limitations of the presented methodologies. The main limitation for discovering decision models using process models as input, is that it can not be fully automated. It often requires the involvement of an expert at some steps of the process. Firstly, when the overlapping patterns are detected in the process models, the expert has to choose which patterns are the best candidate to be included into a dedicated decision models. Secondly, we assumed that the decision activities are those preceding the split XOR gateways, which was not always the case, and the expert involvement is needed to identify and insert the missing decision activities. Our experiments show that the omission of decision activities in the input process models required their insertion at the refactoring phase, which increased the average number of activities in the process models by about 2%. However, this slight increase in process model complexity is fully compensated by a significant simplification of process control flow.

Another limitation occurs during the post-processing of both process and decision models after the externalization of the decision logic. It is recommended that this be done with the guidance of an expert, since in some situation, the domain knowledge is required. One of the firsts step in this direction was done by Wang et al. in [199] by exploring which structures of process model should be presented in the business rules. The extension of this work, e.g., with the concepts represented by the DMN decision models, could be done in future works. Also, the automated adaptation of process and decision labels after model refactoring based on the linguistic analysis [111] could replace the expert work.

8.3 using event log as an input

In this section, we evaluate the methodologies to discover decision models from event logs (cf. Chapters 6–7). Section 8.3.1 presents the implementation details of the research prototype, which discovers decision models using an event log as input. The implementation of the crisp decision model discovery is co-authored by all contributors from [23]. We utilize the presented prototype to conduct a series of the evaluation experiments to discover decision models from event logs. The approach to discover crisp decision models from event logs is evaluated in Section 8.3.2. The evaluation of the methodologies to discover decision models using event log as an input is a novel contribution of this thesis. The approach to discover fuzzy decision models from event logs is evaluated in Section 8.3.3. The implementation of the fuzzy extension to discover decision models from event logs is co-authored by all contributors from [24]. The evaluation of the methodology to discover fuzzy decision models from event logs is based on results published in [24]. Section 8.3.4 provides a summary and a discussion of the presented experiments.

8.3.1 *Implementation*

To conduct experiments on our methodologies for both the crisp and fuzzy discovery of decision models from event logs, we implemented them as respective plug-ins for the ProM framework[2]. ProM is an open-source framework for collecting tools and applications for process mining [188]. The detailed ProM architecture can be found in the documentation provided by Verbeek et al. in [195].

The framework is written in the Java programming language and is extensible by plug-ins. Thus, we extended the

[2] http://www.promtools.org/

existing plug-in *Decision Miner*[3] with the functionality relative to our concepts. To implementing our methodology for discovering crisp decision models from event logs (cf. Chapter 6), we developed the *DMN Extractor* plug-in. The corresponding Java-based implementation and the example log are available at *https://owncloud.hpi.de/index.php/s/dxeGULElonesXxC*. To implement our methodology for discovering fuzzy decision models from event logs (cf. Chapter 7), we developed the *FDMN Extractor* plug-in. The respective Java-based implementation and documentation are available at *https://bpt.hpi.uni-potsdam.de/Public/MiningFuzzyDMN*.

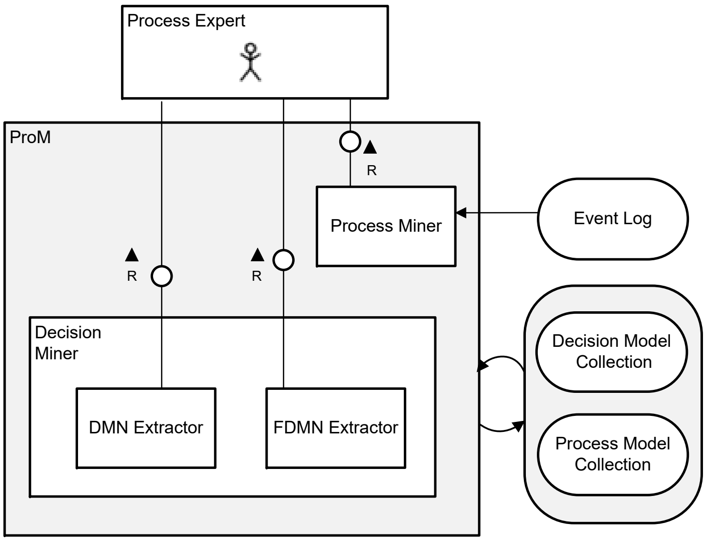


Figure 60: FMC block diagram of the implemented ProM plug-ins to discover both crisp and fuzzy decision models taking an event log as input

The architectural overview of the implemented plug-ins to discover both crisp and fuzzy decision models is presented in a FMC block diagram in Figure 60. The input is the event log, represented in MXML which is an XML-based markup language that is commonly used to implement business logic [56]. When the input event log is imported into ProM, the framework provides multiple possibilities for its analysis. To be able to run the *Decision Miner* plug-in for the discovery of control flow decisions, a process model is firstly discovered with the help of the *Process Miner* plug-in which could be one of the numerous dedicated ProM packages, e.g., the package using the *Alpha++* process mining algorithm [113]. The discovered process model is stored in the *Process Model Collection* storage, as shown in Figure 60.

Once the process model is discovered, the *DMN Extractor* plug-in extracts a DMN decision model, which can be stored in the *Decision Model Collection* storage, as shown in the figure. The *FDMN Extractor*

[3] http://www.processmining.org/online/decisionmining

requires an additional user input for defining the membership functions for the attributes of the generated event log. Once the FDMN model is generated, it can also be stored in the decision model collection. The advantage of using the ProM platform is that further analysis of discovered models can be done with several analysis packages.

Figure 61 shows the user interface of ProM. The screenshot is done at the step, when an input event log is opened. Once the input log is loaded, different ProM functionalities can be applied, such as model mining, analysis, conversion, export, etc.

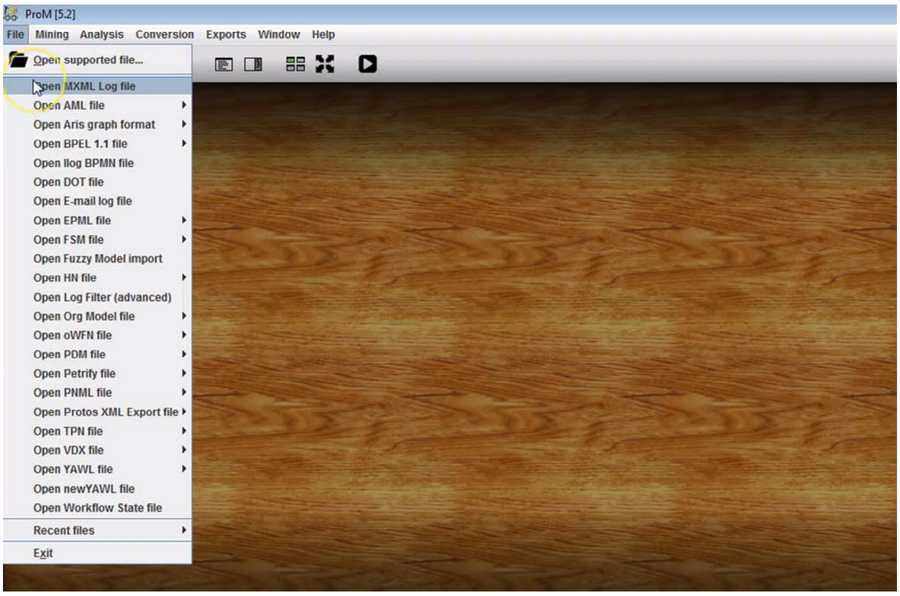
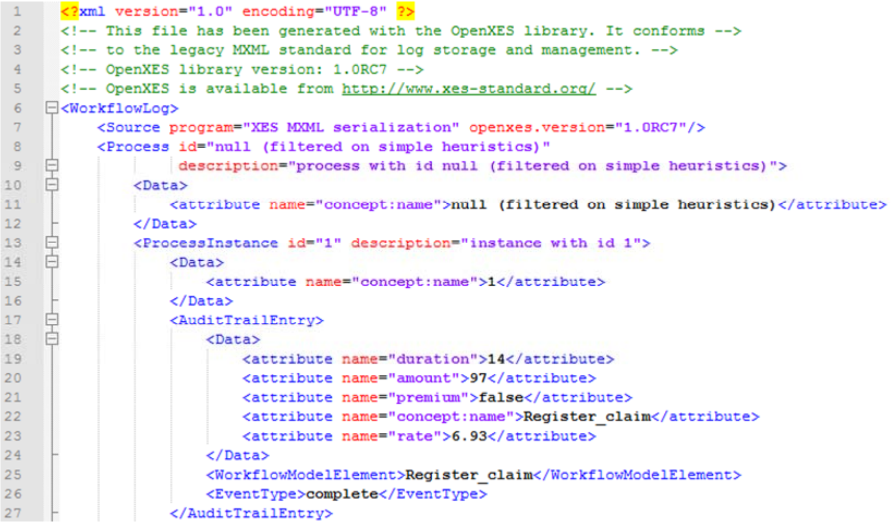


Figure 61: Screenshot of the user interface of the open-source framework ProM 5.2 that we extended with the plug-in to discover DMN decision models from event logs. The input event log can be loaded in various formats. For our experiments, we utilize the MXML log file format.

The work of the developed plug-in is best demonstrated on the example event log which we discussed while conducting the validation experiment in Section 6.5. Thus, we simulated an event log with the help of CPN Tools parameters described in Section 6.5. The screenshot from Figure 62 presents an excerpt of the test event log for the process model from Figure 39 in the MXML file format. It can be seen that the activity instances are recorded in the log in combination with the corresponding data attributes (see the abstract example of the event log in Table 8).

As the next step, a process model corresponding to the event log can be discovered with the help of a process mining algorithm, e.g., the *Alpha++* process mining algorithm [113].

Further, the analysis of the event log is done with the help of the *DMN Extractor plug-in* which outputs the DMN decision model as presented in the screenshot in Figure 63. In the *Decision points* tab,



```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This file has been generated with the OpenXES library. It conforms -->
<!-- to the legacy MXML standard for log storage and management. -->
<!-- OpenXES library version: 1.0RC7 -->
<!-- OpenXES is available from http://www.xes-standard.org/ -->
<WorkflowLog>
    <Source program="XES MXML serialization" openxes.version="1.0RC7"/>
    <Process id="null (filtered on simple heuristics)"
             description="process with id null (filtered on simple heuristics)">
        <Data>
            <attribute name="concept:name">null (filtered on simple heuristics)</attribute>
        </Data>
        <ProcessInstance id="1" description="instance with id 1">
            <Data>
                <attribute name="concept:name">1</attribute>
            </Data>
            <AuditTrailEntry>
                <Data>
                    <attribute name="duration">14</attribute>
                    <attribute name="amount">97</attribute>
                    <attribute name="premium">false</attribute>
                    <attribute name="concept:name">Register_claim</attribute>
                    <attribute name="rate">6.93</attribute>
                </Data>
                <WorkflowModelElement>Register_claim</WorkflowModelElement>
                <EventType>complete</EventType>
            </AuditTrailEntry>
```

Figure 62: Excerpt from the test event log for the process model from Figure 39 in the MXML file format

the plug-in outputs discovered control-flow decisions *p1* and *p3*, and data decisions *risk* and *duration* (cf. Section 6.5). In the center of the screenshot, the output decision model is shown which shows the decision requirements level reflecting the data nodes, discovered decisions, and their dependencies.

The decision logic discovered by our plug-in can be found in both *Decision Tree* and *Decision Tables* tabs, depending on which representation of a decision a user wants to see. In Figure 64, a screenshot of the decision table corresponding to control flow decision *p1* is shown.

The ability of the tool to derive decision models from event logs can also be found in a screencast available at *https://bpt.hpi.uni-potsdam.de/foswiki/pub/Public/WebHome/DMNanalysis.mp4* using the running example from Section 6.5. The screencast reflects our step-by-step approach proposed for the discovery of decisions from event logs, which is described in Section 6.5.

In the next sections, we use the presented implementation to evaluate the impact of our methodologies to discover both crisp and fuzzy decision models from event logs.

8.3.2 *Crisp Decision Model Discovery*

To evaluate our methodology for discovering crisp decision models (cf. Chapter 6), we apply it on a real-life event log. For the evaluation experiments, we consider the road fines event log which was extracted from an information system handling road-traffic fines by the Italian local police [52]. Below, we firstly prepare the log for the application of our decision model discovery methodology. Secondly, in accordance to our methodology (cf. Section 6.2), we

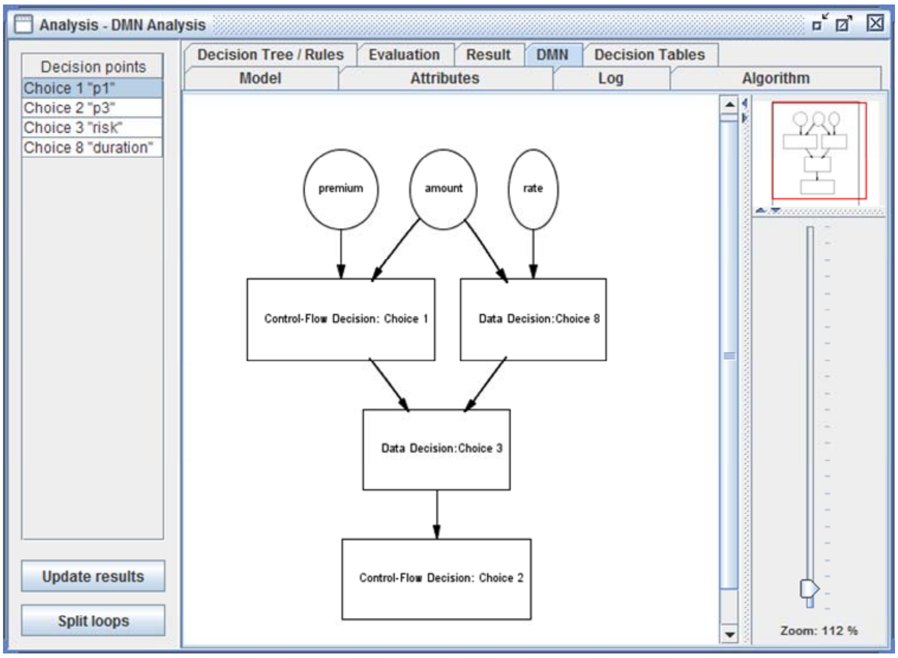
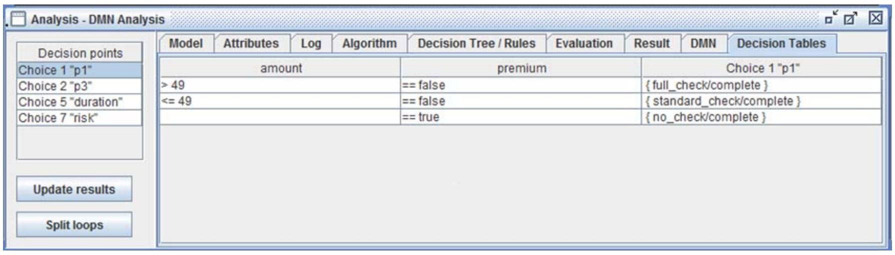


Figure 63: Screenshot of the DMN decision model discovered by our developed plug-in for the event log considered in Section 6.5



amount	premium	Choice 1 "p1"
> 49	== false	{ full_check/complete }
<= 49	== false	{ standard_check/complete }
	== true	{ no_check/complete }

Figure 64: Screenshot of the excerpt from the decision table discovered for the control flow decision *p1* from Figure 39

conduct three steps of the decision model discovery: (1) Discovery of control flow decisions; (2) Discovery of data decisions; and (3) Discovery of decision dependencies. Thereafter, we conduct experiments, measuring the accuracy of the discovered decision model for two event logs: (1) real-life road fines event log; and (2) synthetic loan assessment log considered in detail in Section 6.5.

Preprocessing of the Road Fines Event Log

The considered event log contains 150,370 process instances with 561,470 event instances. The process instances recorded in the log were observed and recorded in the period from 2000 until 2013.The event log contains records for occurrence of 11 activities, namely: *Create Fine, Send Fine, Insert Fine Notification, Insert Date Appeal to Prefecture, Appeal to Judge, Add Penalty, Send for Credit Collection, Send Appeal to Prefecture, Notify Result Appeal to Offender,* and *Payment*. The considered event log is especially suitable for our experiments since it records not only activities, but also the following data attributes: *Amount, Article, Dismissal, Expense, LastSent, Matricola, NotificationType, PaymentAmount, Points, TotalPaymentAmount,* and *VehicleClass*.

In order to prepare the log to be imported into our developed plug-in (see Section 8.3.1), we conducted a preliminary log analysis with the help of the Disco[4] platform that provides extensive possibilities for the filtering and visualization of logs. The log analysis shows that the direct application of our methodology on the raw data would output wrong results, as the log contains a lot of inconsistencies, e.g., incomplete process instances. At the same time, preprocessing of a real-life event log is a standard procedure, required to adapt the log for analysis [85]. Below we summarize the conducted preprocessing steps:

1. Only complete process instances were considered. Without complete information in the event log, the decision rules can not be discovered. Thereby, consulting the process description provided for the data set (cf. [52]), we chose 2 activities that could serve as end event instances (cf. Definition 17) for complete process instances: *Payment* (67, 201 process instances), and *Send for Credit Collection* (58, 997 process instances). In other words, we only chose such cases (process instances) when either the fine was eventually paid or it was sent for a credit collection. Further, we filtered the log with the Disco endpointer filter to remove the process instances which did not end with the execution of any of these 2 activities. Thereby, all the process traces contain the *Create Fine* activity, so it was chosen as a start event for all the traces. The filtered log contains 126, 198 process instances with 498, 816 event instances.

2. Very rare traces in process instances were excluded due to the consideration that if a trace occurs very rarely, it is unlikely to become a rule. Moreover, it is generally not recommended to use machine learning techniques on small sets of input data [205]. Such filtering provides the modified log with 126, 198 process instances and 495, 396 event instances. Up until this point, we preserved 99.95% of all the events in the log.

3. As discussed in our assumptions for the discovery of crisp decision models from event logs (cf. Section 6.2), the decisions in the event log should not appear within loops. Therefore, we applied the Disco filter which eliminates looping subsequences of traces. In particular, the *Payment* activity was recorded several times within a loop, so we filtered the log by eliminating such paths. The event log filtered at this step contains 116, 553 process instances with 437, 380 event instances. It is worth noticing that by executing this

[4] https://fluxicon.com/disco/

filter, we now consider 88.29% of the process instances from the log modified at the previous step. This is a reasonably legitimate threshold [184], and, therefore, this assumption can be considered However, addressing this assumption in the future works, would provide even further enhancements of our presented methodology.

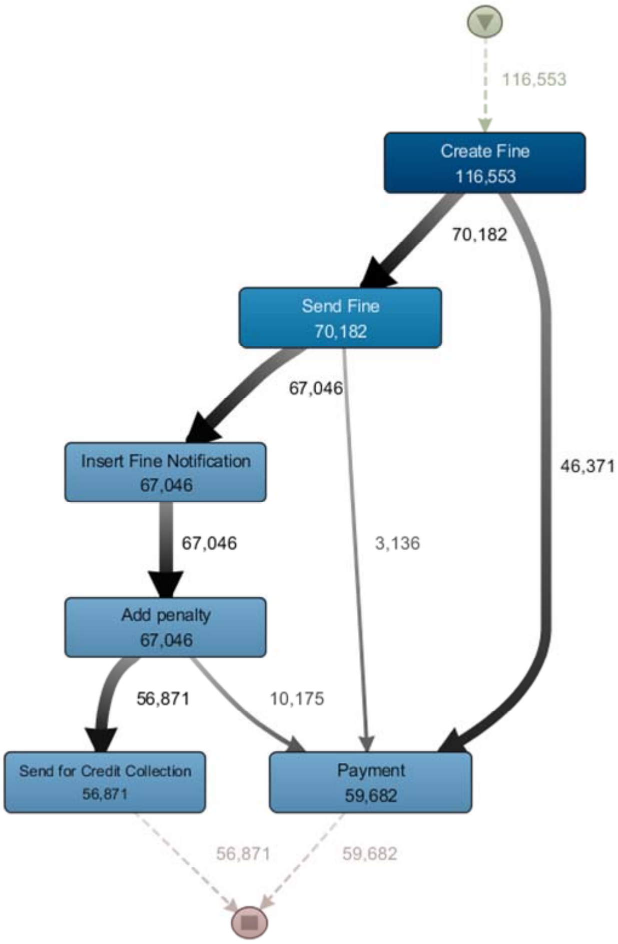


Figure 65: Heuristic net representing the road fines event log, preprocessed for our experiments for the discovery of decision models from event logs

The visualization of the event log obtained by the preprocessing steps described above is presented by a heuristic net in Figure 65. A heuristic net is a directed graph showing causal relations between activities [201]. The numbers in the figure represent the amount of traces corresponding to the causal relations between activities presented in the log.

The entries in the processed log do not explicitly specify the start and end events. In order to utilize the process mining techniques in the most correct way, we artificially specify the start and end events in the filtered log. It can be seen from Figure 65 that, whereas the process model clearly has one start *Create Fine* event, there are several possible end events. From the description of the road fine process provided in [52], it is clear that the *Payment* activity is executed in different contexts. Thereby, relabeling events is a valid procedure that allows a user to explore different representations of the same log [118]. Thus, in accordance with the provided description, we relabel the *Payment* activity that follows directly after the *Create Fine* activity as the *Direct Payment* activity. If the *Payment* activity directly follows the *Add Penalty* activity, we relabel it as the *Payment after Penalty* activity. In case where the *Payment* activity follows the *Send Fine* activity, we keep the original name of the activity. The *Send for Credit Collection* activity represents another possible end event.

The process model shown in Figure 66 shows the BPMN process model of the road traffic management process. We designed this process model manually using the discovered heuristic net from Figure 65, our assumptions on the process activities and events stated above, and the description of the process provided in [52]. The activities of the process model from the figure correspond to activities from the discovered heuristic net. The data nodes of the process model correspond to data attributes recorded in the event log with the corresponding activities.

The process starts with the *Create Fine* activity which records the fine information in the *VehicleClass, Points, Article, Dismissal,* and *Amount* data nodes. This can be directly followed by the *Direct Payment* activity. If this is not the case, the fine is sent to the offender by executing the *Send Fine* activity which incurs certain costs recorded in the *Expense* data node. If the fine is physically handed over to the offender (e.g., by means of a parking ticket), the offender is able to immediately pay the fine which is done by executing the *Payment* activity. In the other case, the offender receives the fine, and the information about this receipt is recorded by the *Insert Fine Notification* activity in the *NotificationType* and *LastSent* data nodes. This is followed by the execution of the *Add Penalty* activity which updates the *Amount* data node. Afterwards, if the offender pays the fine, the *Payment After Penalty* activity is executed. If the offender does not pay, the fine is sent for credit collection (i.e., *Send for Credit Collection* activity). In all cases of payment, the corresponding information is recorded in the data nodes *Payment Amount* and *Total Payment Amount*.

With the help of the DMN Analysis plug-in developed by us (cf. Section 8.3.1), next, we conducted all the steps of our methodology for decision model discovery from Chapter 6.2. The maximal error for the classification results is set to the value of 0.2. Below is the detailed description of the obtained results.

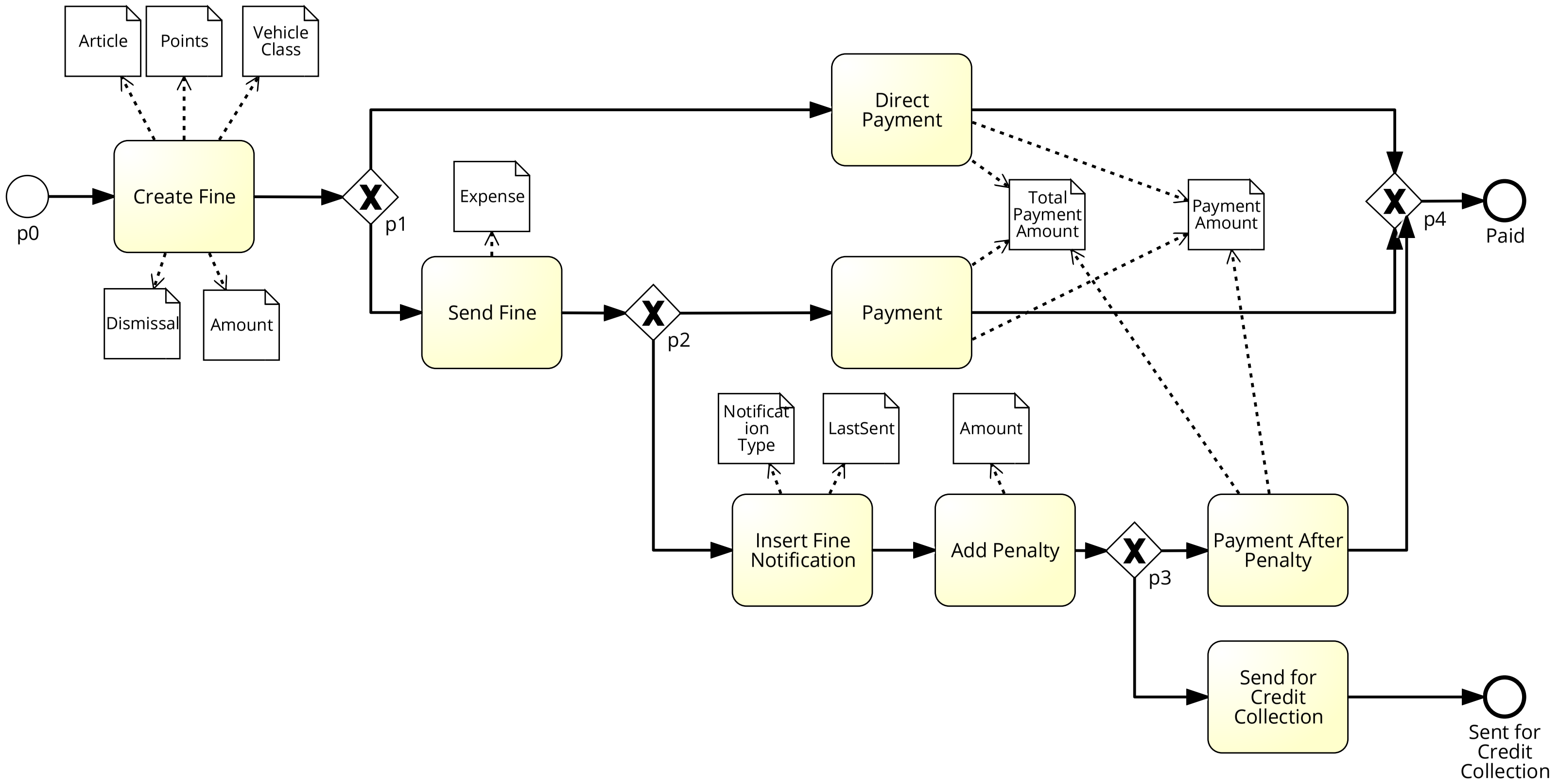


Figure 66: BPMN process model of the road traffic management process

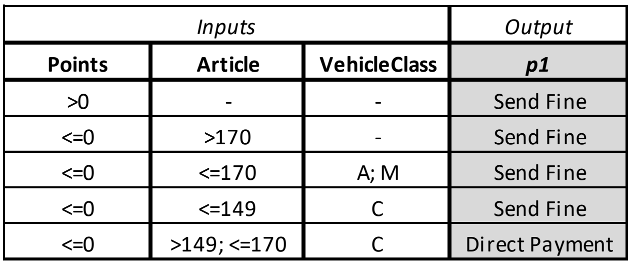
Discovering Control Flow Decisions from the Road Fines Event Log

With respect to our methodology for discovery of control flow decisions from event logs (cf. Section 6.3.1), the program discovers one control flow decision *p1* preceding the *Direct Payment*, or *Send Fine* activities. The decision depends on the *Points, Article,* and *VehicleClass* data attributes. The decision table corresponding to the discovered decision is presented in Table 14:

It is worth noticing that not all the split gateways of the process model (cf. Figure 66) were detected as control flow decisions. Thus, the split gateways *p2* and *p3* only route the process control flow without underlying decision rules. As discussed in Section 5.3, such decisions are managed by process engines and they should not be included into dedicated decision models.

Discovery of Data Decisions from the Road Fines Event Log

Executing the algorithms to discover data decisions from event log (cf. Section 6.3.2), the program did not initially detect any data decisions. Thereby, if a data attribute from the road fines event log would represent a rule-based data decision classifying the influencing attributes with an error less than 0.2, our implementation of Algorithm 1 would have detected it. Therefore, there was only one possible way to try establish whether there are any data decisions in the log, and this was the modification of the part of Algorithm 1 that detects the functional data decisions. The possible reason for the fact that no functional data decision was discovered from the considered event log is that none of the functional operators $+, -, *, /$ used in Algorithm 1 could correctly classify enough of instances from the event log.



Inputs			*Output*
Points	**Article**	**VehicleClass**	***p1***
>0	-	-	Send Fine
<=0	>170	-	Send Fine
<=0	<=170	A; M	Send Fine
<=0	<=149	C	Send Fine
<=0	>149; <=170	C	Direct Payment

Table 14: Decision table corresponding to the *p1* control flow decision discovered from the road fines event log

In order to obtain more insights into potential existence of the other kinds of functional data decisions in the considered event log, we manually conducted the statistical analysis by consequently taking the values of the attributes of the log as classes and the corresponding values of their influencing attributes as training instances.

The analysis was conducted with the help of the Weka library[5]. The results provided by the linear regression models for both discovered functional decisions were yielding the results with the mean square error less than 0.2.

Note that the linear regression form of representation of a data decision adheres to our definition a functional data decision (cf. Definition 20). Therefore, we adapted our implementation of the Line 11 of Algorithm 1 by replacing the $funcs$ function with the linear regression form: $funcs \leftarrow$ {function "$k_0 + k_1 * a_1 + ... + k_z * a_z$" $\mid a_1, ..., a_z \in A_{inf_{num}}$}, where $a_1, ..., a_z \in A_{inf_{num}}$ are attributes influencing the data decision, and $k_0, ..., k_z$ are the regression parameters. This modification of the implementation and parameter determination is done based on the Weka library.

In this new setting, two functional data decisions were detected: (1) *TotalPaymentAmount* with the influencing attribute *PaymentAmount*, and (2) *PaymentAmount* with the influencing attributes *Points* and *Article*. The *TotalPaymentAmount* functional decision represents a linear regression over the *TotalPaymentAmount* data attribute, identified with the mean absolute error 0.12:

$$PaymentAmount = 0.99 * TotalPaymentAmount + 0.11 \tag{22}$$

The *TotalPaymentAmount* functional decision represents a linear regression over the *Article*, and *Points* data attributes, identified with the mean absolute error 0.15:

$$TotalPaymentAmount = -0.02 * Article + 0.11 * Points + 13.70 \tag{23}$$

There exist an abundance of the other non-linear regression functions that can potentially be used to discover the functional data decisions, such as polynomial, hyperbolic, power, exponential, and other types of regression functions [205]. The consideration of such types of functional data decisions are out of the scope of this thesis. However, we demonstrated above, using the example of linear regression, how the other types of functional data decisions can be discovered by adapting Algorithm 1.

Discovery of Decision Dependencies from the Road Fines Event Log

The aggregate of the discovered elements of the DRD diagram is presented schematically in Figure 67. The depicted elements which are used for the construction of the decision requirements diagram: (1) data nodes (*Points, Article, VehicleClass*), (2) data decisions (*PaymentAmount, TotalPaymentAmount*), and (3) control flow decision (*p1*). With regards to the decision logic layer, the program identified a decision table for the control flow decision *p1* (cf. Table 14), and the

[5] http://www.cs.waikato.ac.nz/ml/weka/

corresponding functions for the *PaymentAmount*, and *TotalPaymentAmount* functional data decisions (cf. Equations 22–23)

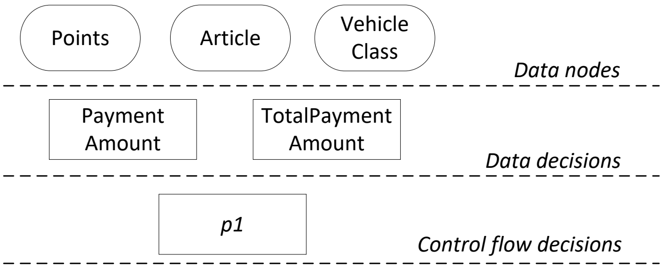


Figure 67: Discovered elements of the DRD diagram for the road fines event log

In the last step of the decision model discovery algorithm (cf. Section 6.2), our plug-in discovers the dependencies between the discovered DRD elements from Figure 67. Thus, the program finds the trivial dependencies between the data decisions *PaymentAmount* and *TotalPaymentAmount*. Also, the information requirements between the discovered decisions and influencing data attributes, represented by the corresponding data nodes, were added to the output DRD model.

The extracted DRD model is presented in Figure 68. The model explicitly shows the decisions corresponding to the process from Figure 66. As it was derived from the event log, it serves as an explanatory decision model for historical decision making and thus, could also serve for compliance checks. If the derived decision model is implemented as a decision service [142], it can be executed complementary to the process model, thereby supporting the principle of separation of concerns (cf. Chapter 2.3.3).

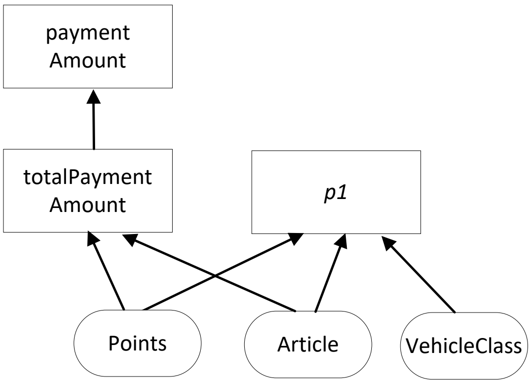


Figure 68: Discovered DRD diagram for the road fines event log

Model Accuracy for the Loan Assessment and Road Fines Event Logs

As far as the evaluation metrics are concerned for assessing the performance of the discovery algorithm, we measure the *accuracy* of the output decision model . In our setting, the decision model accuracy is the average number of event instances that are correctly classified by either decision tables, or functions corresponding to the discovered decisions which constitute the output DRD model, divided by the total number of all instances in the test log.

The statistical and machine learning theories refer to *over-fitting* [205] which is a common methodological mistake to learn and test the performance of a discovered model on the same data as the generated model. The problem with such approach of model discovery is that it might exhibit good accuracy on the training data but perform very poorly on a new set of input data. To avoid over-fitting, the available data should be separated into a training data set that is used to generate the model, and a test data set that is used to evaluate the accuracy of the generated model. The most common approach to do this is called *k-fold cross validation* (cf. [205]), and this is the evaluation method that we use to evaluate the accuracy of the discovered decision models.

With the *k*-fold cross validation, the log is split into *k* approximate equal sized partitions called folds. Each fold is used exactly once as the test set while all the remaining data is used as the training set. The overall performance of a technique is calculated by averaging the performance of that technique on each fold. To evaluate the accuracy of the output decision model, we conducted a commonly used 10-fold cross validation [205] over the considered event log.

It is a common practice to additionally evaluate the behavior of the discovery algorithms by evaluating the accuracy of the model discovered from "noisy logs" as input. Normally, these logs are created by adding incorrect event labellings. Therefore, we introduced noise in the test event log by randomly selecting from 0% to 10% of the event log instances, and randomly replacing their class labels with distinct class labels.

With respect to the introduced noise, the resulted mean accuracy of the discovered models in 10 folds is presented in Figure 69 for the real-life road fines event log considered above. The detailed numerical results of this evaluation experiment can be found in Table 19 from Appendix C. In order to provide a comparison of the classification results with a synthetic setting, we also measured, in the same way, the accuracy of models discovered from the synthetic loan assessment event log which was described in Section 6.5. The detailed numerical results of this evaluation experiment can be found in Table 20 from Appendix C. The run time for discovering the DMN decision model was 0.9 s for the synthetic log, and 2.6 s for the preprocessed real-life log.

As can be seen from Figure 69, our algorithm to discover decision models from event log provides overall a good mean accuracy of 1.00 and 0.84 in the absence of noise for synthetic and real-life logs

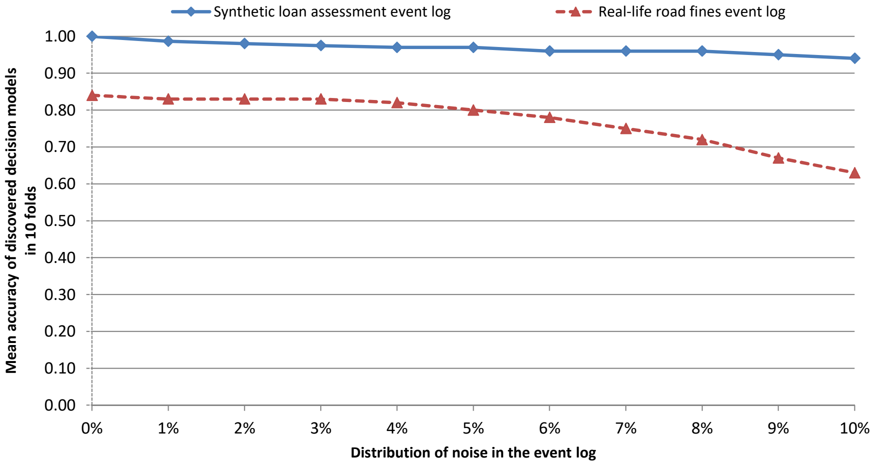


Figure 69: Accuracy of the decision models extracted from both the synthetic loan assessment event log, and from the real-life road fines event log

correspondingly. Naturally, the accuracy of the decision model discovered from the synthetic loan assessment event log is higher, and the algorithm stability is better than for the real-life road fines event log.

For comparison, Baesens et al. in [12] conduct experiments on the decision tree classification by C4.5 algorithm to discover decision tables on three large real-life data sets containing 20, 33, and 33 data attribute inputs, which yield the mean accuracy of 0.81, 0.78, and 0.83 correspondingly. However, these accuracy values can be compared with our results only in an approximate fashion, since the accuracy of the output decision model from our approach is calculated as a mean of the values of all the discovered decision logic structures associated with the decisions in the output DRD. As it also pointed out in the related work in Section 3.5.2, the discovering of decision models complementary to process models from event logs has not been explored in the related works yet, so to the best of our knowledge, there are no other experimental results based on which we could provide a more detailed analysis of the accuracy of the output decision model. At the same time, future works could consider the application of other classifiers to discover decision models from event logs, and the comparison of the model accuracy could be done, e.g., by adapting the SVM-based decision discovery [205] to our problem.

8.3.3 *Fuzzy Decision Model Discovery*

For the evaluation of our methodology for decision model discovery extended with fuzziness (cf. Chapter 7), we conducted the evaluation experiments on a credit-risk assessment process from Section 7.4.

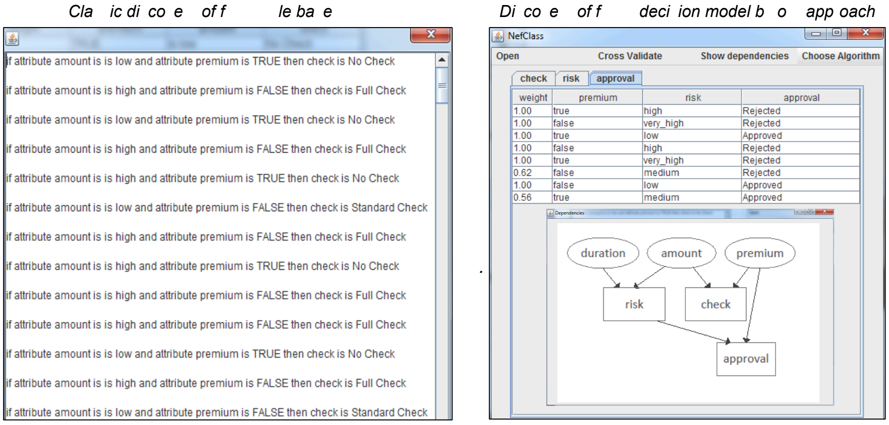
As was stated by Assumption 5 in the approach description in Section 7.3.1, in order to conduct the discovery of a fuzzy decision model from an event log, there should exist a mechanism that describes numerical attributes of the input event log in linguistic terms. This should be done by a process expert, since domain knowledge is required. The involvement of a process expert at this stage was not possible, and therefore, we considered the synthetic credit-risk assessment event log described in Section 7.4 for the evaluation experiment. Although the event log is simulated, it represents a real-world context as it was simulated by us based on our knowledge of its real-world parameters from [12].

Experimental Setup

For the test data, we considered the process of a loan assessment from Section 7.4 and processed an event log consisting of 1000 process instances, modeled close to real data from [12] with the help of the simulation software, CPN Tools [3]. The simulation parameters are presented in Section 7.4. The generated log contained two numerical attributes, *duration* and *amount*, and two nominal attributes, *premium* and *risk*. The assigned linguistic terms and membership functions for *amount* and *duration* can be seen in Figure 47. With the help of our implementation (cf. Section 8.3.1), a corresponding DMN decision model was discovered, as described in detail in Section 7.4. For fuzzy rules corresponding to decisions in the output DRD model which was obtained by the GA classifier, the maximal rule error was set to 0.2, the minimal coverage was set to 0.42, and the minimal fitness was set to 0.1. The minimal fitness for NF was also set to 0.1. We ran the GA classifier until the number of low-performing rules, that were not added to the output rule sets corresponding to process decisions, reached a threshold of 50000 rules. The NF classifier was running until the entire event log was processed. Both algorithms also stopped if the output decision table was *complete*, which means that for all the combinations of input values there was a rule that covered it. The average run time for discovering FDT per process decision was 44.2 s for the GA classifier, and 1.8 s the for the NF classifier.

Interpretability of the output FDMN

A screenshot of the FDMN, which our application outputs for the input log, is presented on the right side of Figure 70. Corresponding to each decision in the DRD are the FDTs, which are obtained by the NF classifier, and post-processed according to our methodology. In the left sreenshot one can see the direct application of the "state-of-the-art" NF classification for process decisions *check*, which yields a



Cla ic di co e of f le ba e

if attribute amount is is low and attribute premium is TRUE then check is No Check
if attribute amount is is high and attribute premium is FALSE then check is Full Check
if attribute amount is is low and attribute premium is TRUE then check is No Check
if attribute amount is is high and attribute premium is FALSE then check is Full Check
if attribute amount is is high and attribute premium is TRUE then check is No Check
if attribute amount is is low and attribute premium is FALSE then check is Standard Check
if attribute amount is is high and attribute premium is FALSE then check is Full Check
if attribute amount is is high and attribute premium is TRUE then check is No Check
if attribute amount is is high and attribute premium is FALSE then check is Full Check
if attribute amount is is high and attribute premium is FALSE then check is Full Check
if attribute amount is is low and attribute premium is TRUE then check is No Check
if attribute amount is is high and attribute premium is FALSE then check is Full Check
if attribute amount is is low and attribute premium is FALSE then check is Standard Check

Di co e of f deci ion model b o app oach

weight	premium	risk	approval
1.00	true	high	Rejected
1.00	false	very_high	Rejected
1.00	true	low	Approved
1.00	false	high	Rejected
1.00	true	very_high	Rejected
0.62	false	medium	Rejected
1.00	false	low	Approved
0.56	true	medium	Approved

Figure 70: Screenshots of the results from our prototype

large fuzzy rule base that is difficult for humans to interpret. In contrast, FDMN, derived by our methodology, (in the right screenshot of Figure 70) consists of compact FDTs incorporating rule weights and fuzzy hit policies, and it shows dependencies between decisions. It has to be noted that the interpretability of the output FDMN highly depends on the fuzzy algorithm applied for learning the model. According to our observations, NF often leads to a smaller amount of rules than GA.

Accuracy of the output FDMN

To evaluate the accuracy of the output FDMN, we again used the 10-fold cross validation over the test log [205]. Here, the accuracy is equal to the average number of event instances that are correctly classified by the rules in FDTs divided by the total number of all instances. Further, in order to introduce a noise in the input event log, we again created incorrect log entries by randomly selecting from 0% to 10% of the event log instances, and randomly replacing their class labels with distinct class labels. The resulted accuracy of the classifiers with respect to the introduced noise is presented in Figure 71. The detailed numerical results of this evaluation experiment can be found in Appendix C: the classification results obtained by the application of the neurofuzzy classifier are presented in Table 21, the classification results obtained by the application of the genetic classifier are presented in Table 22.

Both classifiers achieved high accuracy for the event log without noise. Further, it can be seen, that NF shows good stability, as it preserves a very good accuracy of ca. 90.21% in the presence of 10% input data noise. GA shows less stability, as its accuracy reduces to ca. 82.10% in the presence of 10% input data noise. Setting the algorithms parameters can be used to adjust the desired user's output. For example, if a smaller amount of rules for better human interpretabil-

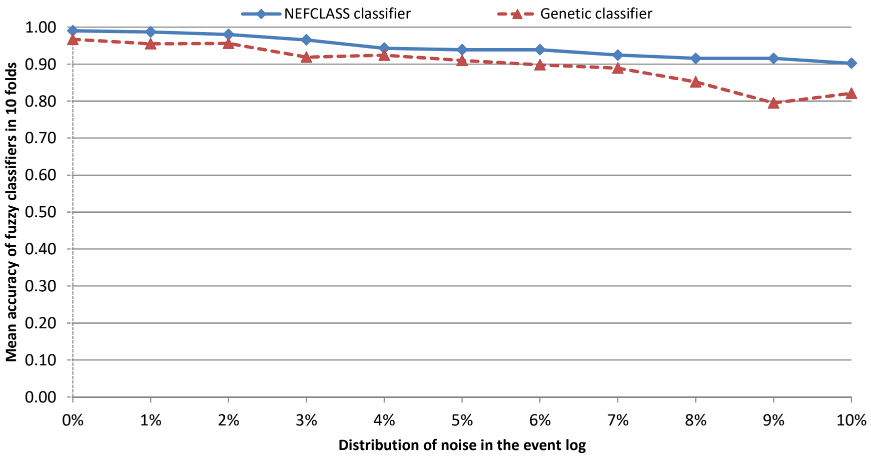


Figure 71: Mean FDMN accuracy in 10 folds with respect to introduced noise in the log

ity is needed, the threshold has to be higher than 0.1 used in our case, although the accuracy might decrease. Further experiments on evaluating the accuracy with regards to tuning of termination criteria are planned for future work.

8.3.4 *Summary and Discussion*

The considered perspective in the section above is the discovery of decision models using event log as an input. This perspective incorporates both crisp decision model discovery presented in Chapter 6 and its fuzzy extension presented in Chapter 7.

The first evaluation method applied by us is the design and implementation of a prototype validating the presented methodology. The Prom framework is an open-source platform that is designed for collecting tools and applications of discovering process models from event logs. We extended the *Decision Miner* plug-in of ProM for crisp decision model discovery, and developed an extension of it with fuzziness, both of which we use throughout our evaluation experiments.

The second evaluation method that we use is the investigation of how the developed methodologies interact with the real-world context. The presented methodologies to discover decision models using event log as an input are based on solving the classification problem, where the classes correspond to process decisions, and the training examples are the process instances from the event log. Therefore, we use the accuracy of the output decision model as the main evaluation metric. The evaluation of the fuzzy decision model discovery required measuring the interpretability of the output decision model as an additional evaluation metric. Also, the evaluation of the discovery of a crisp decision model, and of a fuzzy decision model from an event log required different types of the real-wold context.

In order to do conduct the crisp decision model discovery, we conducted a series of experiments on the real-life road fines event log. In order to apply our methodology for discovering a decision model, the event log has to be preprocessed by: (1) consideration of complete process instances; (2) exclusion of rare traces which are unlikely to contain decision rules; (3) excluding looping subsequences of traces; and (4) adding artificial start and end events as the log does not explicitly specify such events. This preprocessing goes in line with our assumptions on the presented approach (cf. Section 6.2).

The application of our methodology to derive crisp decision model from the road fines event log preprocessed in the above mentioned way was successful and yielded a valid DMN decision model explaining the historical decision making, recorded in the input event log. A major insight obtained from the application of our approach to derive crisp decision model from the test log was concerning the derivation of functional data decisions. Thus, in the initial example implementation of our methodology to discover functional data decisions (cf. Algorithm 1), we assumed only four arithmetical operators that could be used in order to detect corresponding data decisions. However, our manual analysis of the road fines event log showed that some of the data attributes represent a linear regression over the other data attributes from the log.

Although any type of a functional dependency was foreseen by our theoretical framework (cf. Definition 20), we had to adapt our implementation. This was done by considering functional dependencies between the data attributes in the log, and then the output decision model, which additionally contained the functional data decisions that represented linear regressions of one data attribute over the other data attributes from the log. However, there can exist many other types of non-linear regression functions that can also be used to discover the functional data decisions, e.g., polynomial, or exponential. The implementation of the discovery of these types of functional dependencies are out of the scope of this thesis, but future works could consider such implementations and corresponding experiments to find the most suitable type of functional dependencies in case several of them can be detected for the same data attributes in the input event log. Currently, for conducting this step we recommend the involvement of a process expert which is a limitation of the presented methodology.

With regards to the accuracy of the output decision model, the application of our methodology on the road fines event log yield the result value of about 84% which is a good result compared to the state-of-the-art techniques to mine decision rules. In order to test stability of the algorithm, we introduce a noise level from 0% to 10% in the input event log, and conducted a 10-fold cross validation of the training instances. Even in the presence of 10% of noise in the test log, the mean accuracy of the output decision model has the value of about 64% which again is good result compared to state-of-the-art and which demonstrates that the presented algorithm is quite stable.

With respect to the fuzzy extension of the decision model discovery methodology, we could no longer use the road fines event log for the evaluation. In order to conduct the fuzzy decision model discovery, we would have to involve an expert who would describe numerical attributes of the input event log in linguistic terms (cf. Assumption 5 in Section 7.3.1). Since the process expert involvement was not possible, for the evaluation experiments we considered the syntethic credit-risk assessment event log described in Section 7.4 as it we simulated by us based on our knowledge of its real-world parameters from [12].

The application of our methodology to derive fuzzy decision model from the credit-risk assessment event log was successful and yielded a valid fuzzy DMN decision model. To evaluate its interpretability, we also conducted the state-of-the-art discovery of a set of fuzzy rules corresponding to the test event log. Our results show that state-of-the-art techniques for mining fuzzy decision rules provide a large fuzzy rule base that is difficult to be interpreted by humans. In contrast, the decision model derived by our methodology provides the user with compact fuzzy decision tables, shows dependencies between decisions, and it incorporates fuzzy hit policies.

A major insight obtained from the application of the fuzzy decision model discovery from the test event log is that choosing the criteria for the termination of the discovery algorithm can be challenging, and highly depends on the requirements of the business environment. In our experiments, both genetic and neurofuzzy classifiers terminated their work when the output decision table was complete, which means that for all combinations of input values there was a rule that covered it. However, other termination criteria could be used, e.g., reaching certain rules or other criteria considered by us in Section 7.3.4. Therefore, choosing the best suitable termination criteria should be done by a stakeholder who is interested in applying our methodology.

With regards to the accuracy of the output fuzzy decision model, we conducted the same experiment of testing it against introduction of the noise from 0% to 10% and measuring the mean accuracy with the 10-fold cross validation. Both classifiers show almost perfect accuracy of more than 99% in the absence of noise. The neurofuzzy classifier shows a better stability, as it preserves a very good accuracy of more than 90%, whereas the resulting accuracy provided by the genetic classifier goes down to about 82%. However, setting the parameters of classifiers can also be used to adjust the input which would be the most beneficial to the business environment. For example, if a smaller amount of decision rules is more preferable, the minimal rule fitness should be set higher than 0.1, which is used in our experiments. On the other hand, that might decrease the accuracy of the output fuzzy decision models. Future works can consider finding a desired balance between accuracy and interpretability of the output fuzzy decision models.

The application of both the genetic and the fuzzy classifier revealed the following limitations of our methodology. Since the discovery of fuzzy rules requires a predefined membership functions and linguistic literals, the discovery accuracy depends on the quality of those literals. If the membership functions are inaccurate, the number of misclassified instances in the training data increases. That is why in most cases the definition of the membership functions has to be done by an expert. Otherwise, it can lead to a higher probability of wrong rules which has a negative impact on the accuracy of the discovered decision model.

8.4 CHAPTER SUMMARY

The main purpose of this chapter is to evaluate the introduced methodologies for the discover decision models complementary to process models. As the first evaluation method, we chose to design and implementation prototypes to perform the evaluation. As the second evaluation method, we evaluated the application of the developed methodologies in the real-life context. Thereby, two perspectives of decision model discovery were considered. Section 8.2 presents the evaluation of the methodologies to discover decision models using process model as an input. Section 8.3 considers the evaluation of the methodologies to discover decision models using event log as an input. We presented the evaluation summary, and discuss the obtained insights with respect to these two perspectives. For each discovery perspective, we presented the corresponding implementation, detailed steps performed during the experiments, an analysis of the results obtained, the limitations of the methodologies, and suggestions for future work.

In conclusion, we showed that all of the approaches are able to derive a valid output decision model. In cases where process models are used as an input, our pattern-based-discovery shows high occurrence frequency of the developed patterns in real-life process model repositories. Our approach also allows us to reduce the complexity of input process models by the simplification of process model structures that are externalized in a dedicated decision model. In cases where an event log is used as an input, both crisp and fuzzy approaches to discover a decision model demonstrate high accuracy and stability. A common drawback for all methodologies is that the involvement of a process expert is needed for controlling some steps of the decision model discovery. Apart from that, preprocessing of the event log, and post-processing of the output models is needed.

The evaluation results and insights provide input for a roadmap which can be used for future research that can further enhance the discovery of decision models complementary to process models, which we discuss in the next chapter.

9 CONCLUSION

This last chapter serves as a conclusion for the presented thesis. Section 9.1 summarizes the thesis results, i.e., the formal framework for complementary process and decision modeling, the methodologies to discover decision models using process models as input, and the methodologies to discover decision models using event logs as input. In Section 9.2, we discuss limitations of our work and highlight opportunities for future work on the discovery of decision models complementary to process models.

9.1 THESIS RESULTS

Business Process Management (BPM) offers competitive benefits for organizations, that take advantage of BPM to run the day-to-day processes of the organization, in both the public and private sectors. Process modeling is an essential stage of BPM projects. Process models are a tool used to design and execute business processes. In the course of BPM projects and organizational activities, situations frequently occur, where it is required to make a choice between several alternatives. The choice made can have a significant impact on the future direction of a process and even an entire organization. Thus, decision making plays an important and vital role in such situations, making it an important component in any organization. The need for making effective decisions is clear and decision models can complement process models to assist with this. Decision models are used to model and execute decision making in business processes.

Recently, integrated frameworks that provide support for modeling and execution of both processes and its decision making have been appearing. Complementary business process and decision management improve the process by both focusing on the way decisions are made and directing the processes that incorporate decision making. However, modeling and executing decisions complementary to processes is a challenging task, most due to the fact that the separation of process and decision logic is not straightforward in real-life business processes. At the same time, the existing works lack guidelines on how to achieve the separation of concerns in complementary business process and decision management.

In this thesis, we tackle this problem by introducing a set of concepts and methodologies to discover decision models complementary to process models. To achieve this, we make use of the information about the process decision making which exists in companies, such as process models and event logs. After an extensive research process, the results achieved in this thesis can be summarized as follows.

(1) Identification of the formal specification of the elements and the relationships that form a decision model, which can be designed and executed complementary to the process model.

The major part of this contribution is presented in Chapter 2. In particular, we introduced a novel formal framework for decision-aware process modeling, and process-aware decision modeling. Additionally, the formalisms connected to the execution of processes and decisions, such as process and decision instantiation, and event logs, were introduced. Furthermore, considering that real-life decision processes are often exposed to imprecision, we extended our base formal concept with the notion of fuzziness of decision models in Chapter 7. In Sections 3.1–3.3 of Chapter 3, we related the state-of-the-art works to the formal specification for complementary process and decision modeling presented in this thesis.

The introduced specification denotes abstract concepts that are independent of modeling notations used by companies, and therefore, can be reused and adapted by stakeholders for modeling decision models complementary to process models with respect to their information systems and notations. At the same time, all the introduced formal specifications are illustrated with the help of graphical constructs by utilizing BPMN for process modeling, and DMN for decision modeling.

(2) Development of the methodologies to extract decision models from control flow and data of process models.

The methodology to extract decision models from the control flow of process models is presented in Chapter 4 of this thesis. The methodology to extract decision models from control flow of process models is presented in Chapter 5 of this thesis. Both methodologies represent semi-automatic approaches to: (1) Identify either control-flow-based (Chapter 4) or data-centric decision patterns (Chapter 5) correspondingly in a given process model; (2) Derive a corresponding decision model; and (3) Refactor the original process model by replacing the decision logic accordingly. The identification is pattern-based, and is derived from an intensive analysis of more than 1000 real-world process models. In Section 3.5.1 of Chapter 3 we review other approaches that attempt to discover decisions from process models, and provided an extensive comparison of them with our work.

We implemented our methodologies and provided statistical insights about pattern utilization in the real-world process models, obtained from several different industries. Overall, the patterns occurrence in the considered repositories demonstrate that the identified patterns are frequently used in the real-world. Therefore, the usage of the considered patterns as a base to discover decision models complementary to process models is valid and provides a high potential for application in practice.

Based on the detected decision patterns in the real-world process models, we were able to refactor more than two thirds of all the considered process models by externalizing the decision logic into dedicated decision models which can be used complementary to process models. At the same time, it was confirmed, with the help of the methodologies developed by us, that it is feasible to bring into practice the principle of separation of process and decision concerns. The demonstrated reduction of process model complexity indicates that the provided methodologies can serve as tools for business process improvement.

(3) Development of the methodologies to extract crisp and fuzzy decision models from event logs.

The methodology to extract crisp decision model from an event log is presented in Chapter 6 of this thesis. The extension of it is the methodology to extract fuzzy decision model from an event log which is presented in Chapter 7. The introduced methodologies extend an existing approach to derive control flow decisions from event logs, with the additional identification of data decisions, their dependencies, and fuzzy concepts. The methodology to derive a crisp decision model from an event log is based on the semi-automatic decision tree classification of process instances in the log. The methodology to derive fuzzy decision model from an event log is based on the semi-automatic fuzzy classification, whereby the genetic and neurofuzzy classifiers are applied. Section 3.5.2 of Chapter 3 reviews the other approaches to discover decisions from event logs, and compares them to our work.

We implemented the presented methodologies and provided statistical insights for the derivation of both crisp and fuzzy decision models from real-life and simulated event logs. With regards to the classification accuracy of the output decision model, the application of our methodologies on the test event logs yield good result in comparison to the state-of-the-art techniques used to mine decision rules. The proposed algorithms were tested for stability by introducing noise into the input event logs, and compared the results against the state-of-the-art, which showed that the algorithms demonstrate good stability. Additionally, we were able to demonstrate that the interpretability of the output fuzzy decision model, derived by our methodology, is higher compared to the results obtained by traditional fuzzy rule mining.

To sum up, the application of all the developed methodologies to derive decision models complementary to process models, using both process models and event logs as inputs, yielded valid decision models, explaining the historical decision making which was recorded in the input process models and event logs. Thus, it can be stated that the stakeholders are provided with tools for the semi-automated design of decisions, complementary to processes, utilizing the knowledge about "as-is" process decision making. As the output decision models explains the existing or historical decision making at the organization, the application of the presented methodologies can be used, for example, for compliance checks among other uses which could improve the organizations decision making. Additionally, the constructed decision models serve as blueprints for executing decisions complementary to process models, which support the separation of concerns principle.

9.2 LIMITATIONS AND FUTURE WORK

While this thesis has provided the groundwork for the discovery of decision models complementary to process models, there are also limitations of our work. This section summarizes the limitations that have been found during our evaluation and gives an outlook on potential research directions for future work.

Alignment of our Formal Constructs with BPMN and DMN

We chose to base the presented methodologies on the formal constructs which were aligned with BPMN for process modeling, and DMN for decision modeling. By choosing BPMN process models, we faced the challenge related to detecting the decision activities in process models as the standard does not allow explicit representation of the decision activity type, e.g., the user activities can denote decision activities but do not have to. Thus, we assumed that the decision activities are those preceding the split XOR gateways, which was not always the case, and the expert involvement was needed to identify and insert the missing decision activities. This limitation could be overcome if the process model notation utilized by a stakeholder can unambiguously denote if an activity is of a decision type. Alternatively, the activity label analysis [111] can be applied in order to detect the decision activities.

Expert Involvement

A common limitation, occurring in all of the presented methodologies to discover decision models, is that they are semi-automated, since the involvement of an expert at some of the steps is required.

With regards to the methodologies for the discovery of decision models using process models as input, when the overlapping patterns are detected, the expert has to choose which patterns are the best candidate to be included into a dedicated decision model. Secondly, as mentioned above, we assumed that the decision activities are those preceding the split XOR gateways, which is not always the case, and the expert involvement is needed to identify and insert the missing decision activities.

Also, our methodologies assume that the post-processing of both process and decision models after the externalization of the decision logic is done by an expert, as hereby, domain knowledge is required. One of the firsts step in this direction was done by Wang et al. in [199] by exploring which structures of process model should be presented in the business rules. An extension of this work, e.g., with the concepts represented by the DMN decision models, could be a solution to this constraint. Also, the automated label adaptation after model refactoring based on the linguistic analysis [111] could potentially replace the expert work.

Implementation Limitation

Another limitation on the input event log for our methodologies to discover decision models form event logs is rather related to the methodology implementation. Although any type of a functional dependency was foreseen by our methodology to discover decision models from event logs (Chapter 6), as a result of application of it to the real-life road fines event log, we had to adapt our implementation. This was done by considering functional dependencies between the data attributes in the log, and then the output decision model, which additionally contained the functional data decisions that represented linear regressions of one data attribute over the other data attributes from the log. However, there can exist many other types of non-linear regression functions that can also be used to discover the functional data decisions, e.g., polynomial, or exponential. The implementation and experimentation on the discovery of these types of functional dependencies are out of the scope of this thesis. Future works could consider such implementations and corresponding experiments to find the most suitable type of functional dependencies in case several of them can be detected for the same data attributes in the input event log. Currently, for conducting this step we recommend the involvement of a process expert.

Assumptions on the Input Process Models and Event Logs

In our methodologies to discover decision models from process model we assume that input process models are structurally sound,

i. e., they contains exactly one start and one end event and every node of the process model is on a path from the start to the end event. This assumption is reasonable, as usually models exhibiting deadlocks, or livelocks are assumed to be subject to modeling errors [61].

For the methodologies to discover the decisions from event logs, we also assumed that an input event log corresponds to a process model which is structurally sound. Moreover, we assumed that the decisions do not appear within loops as this would result in several instances of one decision within one process execution which would need special handling. The DMN standard, which we use as a base for decision modeling, does not support the design of looping decisions. Therefore, we did not have it as a goal to incorporate decision looping in the outcome decision model. However, it is possible to have situations where the process model handles the decision looping with the help of the control flow structures. This would require special handling, which future research could explore.

Balance between Accuracy and Interpretability

This limitation refers to the methodologies to discover decision models from event logs which are based on solving the classification problems (cf. Chapters 6–7). The tension between classification accuracy and interpretability is a fundamental trade-off question when using machine learning algorithms for prediction [205]. Simpler predictive algorithms, which are more easily interpreted by humans, tend to be less accurate than more advanced methods that are not as easily explained. Further experiments on the parameters of classifiers for the discovery of decision models from event logs could be conducted in future works since we have already developed the plugin which allows this. Thus, the task can be set to find the optimal input which would be the most beneficial for the business environment. For example, if a smaller amount of decision rules is more preferable for the output decision model, the minimal rule fitness over the input event log should be set higher. But on the other hand, that might decrease the accuracy of the output decision models. Future works can consider finding a desired balance between accuracy and interpretability by using our settings to discover both crisp and fuzzy decision models from event logs.

Further Enhancements of Decision Model Discovery

The more information we know about the process, the more precise the decision model extracted from this information could be. In such a way, our approaches to discover decision models complementary to process models from process-related information can be further enhanced.

An important process characteristic is its performance. The performance of a business process can be measured with the help of *key process indicators (KPIs)* [63, 95, 134, 157]. In the context of decision making in the business processes, often KPIs reflect the strategic information, like, for example, knowledge about the time or risks associated with alternative actions to be taken.

The factors that can be considered as performance indicators are domain specific. For example, in the credit-risk assignment we refer to credit scoring quality measures for consumer credit applications presented by Leonard in [109]. Such additional factors can be considered to enhancing our presented approaches to further discover decision models complementary to process models.

Additional extensions of our work could include context-aware analysis of process executions for aiding process decisions. For example, context-aware analysis of event logs for aiding resource allocation decisions that are presented in [175], could be used for the enhanced mining of decision models. In this thesis, we do not consider process context data in our analysis, but in future works, such analysis could be done.

Another direction for enhancement of decision model mining could be based on the translations of natural text expressions, e.g., which exists at enterprises in the form of textual documents, into decision models. One of the first steps in this direction was done in [101] where DMN models are derived from SVBR rules. In [39], more than sixty formally grounded business rule patterns are presented that could be used for expressing decision logic in DMN models, and then corresponding decision mining techniques should be provided. Further metrics such as the reliability of data and the trust and reputation of the data provider [93] could be used in order to preprocess data before extracting it into a decision model complementary to a process model.

According to Rosenblueth and Wiener [163], "no substantial part of the universe is so simple that it can be grasped and controlled without abstraction." Taking into account that the work presented by us is one of the first in the direction of discovering decision models complementary to process models, it was inevitable that even an extensive study, such as that conducted in this thesis, would contain limitations. Nevertheless, this research and associated limitations provide a contribution to the research community for future work, which could investigate and propose solutions to these limitations and challenges.

Part IV

APPENDIX

A

FORMAL DEFINITIONS OF CONTROL-FLOW-BASED DECISION PATTERNS

Below we introduce the formal definitions of three control-flow-decision patterns, which were introduced by us in a semiformal way in Chapter 4 (cf. Section 4.3). All three patterns represent process fragments (cf. Definition 8) that can be observed in a process model (cf. Definition 7). We use subscripts, e.g., N_{pm}, or N_{pf}, to denote the relation of sets and functions to process model pm or process fragment pf and omit the subscripts where the context is clear.

Pattern P1 is a process fragment representing a split gateway with at least two outgoing control flow edges. Considering that given is a process model pm , we define pattern P1 as follows.

Definition 28 (Pattern P1). Let pf be a process fragment of process model pm and let Σ_{pf} denote the conditions assigned to control flow edges. Then, pf represents *P1* if

- $|G_{pf}| = 1 \wedge |g \bullet| \geq 2 \wedge (\eta_g(g) = XOR \vee \eta_g(g) = IOR), g \in G_{pf}$ (the fragment contains exactly one split gateway);
- $|T_{pf}|$ = $|g \bullet| + 1$ (the number of activities of pf equals the number of outgoing edges[1] of the split gateway g plus 1);
- $\bullet g = da \wedge |\bullet g| = 1$, $da \in T_{pf}$ (decision activity da is the only predecessor of the split gateway),
- $|\bullet da| = 0$ (decision activity da is the start node of pf);
- $\forall t \in T_{pf} \backslash da : \bullet t = g$ (all activities other than the one preceding the split gateway g directly succeed g);
- $\forall t \in T_{pf} \backslash\ da : |t \bullet| = 0$ (all activities other than the one preceding the split gateway g are end nodes of pf); and
- $\forall t \in T_{pf} \backslash da\ , c \in C_{pf}$ such that $(g, t) = c : \sigma(c) \in \Sigma_{pf}$ (all outgoing edges of the split gateway are annotated with a condition). $\diamond$

Pattern P2 is a process fragment representing a sequence of split gateways that represents a decision tree. We consider that given is a process model pm. Then, we define pattern P2 as follows.

Definition 29 (Pattern P2). Let pf be a process fragment of process model pm and let Σ_{pf} denote the conditions assigned to control flow edges. Then, pf represents *P2* if

- $\forall g \in G_{pf} : |g \bullet| \geq 2 \wedge (\eta_g(g) = XOR \vee \eta_g(g) = IOR)$ (all gateways of the fragment are split gateways);

[1] the number of outgoing (incoming) edges directly translates to the number of direct successors (predecessors) and vice versa

- $\exists da \in T_{pf} : |\bullet da| = 0 \wedge \forall\, t\in T_{pf} \setminus da : |\bullet t| = 1$ (decision activity da is the start node of pf);
- $\forall t \in T_{pf} \setminus da : |t \bullet| = 0$ (activities other than the start node da are end nodes of pf),
- $\forall g \in G_{pf} : \forall n \in g\bullet : n \in T_{pf} \cup G_{pf}$
  (all successors of a gateway are an activity or a gateway); and
- $\forall t \in T_{pf} \setminus da, g \in G_{pf}, c \in C_{pf}$ such that $(g,t) = c \vee (g,g) = c$: $\sigma(c) \in \Sigma_{pf}$ (all outgoing edges of a split gateway are annotated with a condition). ◇

Pattern P3 is a process fragment representing a sequence of split gateways separated by an activity. Considering that given is a process model pm , pattern P3 is defined by as follows.

Definition 30 (Pattern P3). Let pf be a process fragment of process model pm and let Σ_{pf} denote the conditions assigned to control flow edges. Then, pf represents *P3* if

- $\forall g \in G_{pf} : |g \bullet| \geq 2 \wedge (\eta_g(g) = XOR \vee \eta_g(g) = IOR)$ (all gateways of the fragment are split gateways),
- $\exists da \in T_{pf} : |\bullet da| = 0 \wedge \forall t \in T_{pf} \setminus da : |\bullet t| = 1$ (decision activity da is the start node of pf),
- $\forall t \in T_{pf}$ such that $|t \bullet| = 1 : t \in T_{pf}$ (all activities being no end node are a task),
- $\forall g \in G_{pf} : \forall n \in g\bullet : n \in T_{pf} \cup G_{pf}$ (all successors of a gateway are an activity or a gateway),
- $\forall n \in N_{pf}$ such that $|n \bullet| = 0 : n \in T_{pf}$ (all end nodes are activities),
- $\forall t \in T_{pf} \setminus da, g \in G_{pf}, c \in C_{pf}$, such that $(g,t) = c \vee (g,g) = c$: $\sigma(c) \in \Sigma_{pf}$ (all outgoing edges of a split gateway are annotated with a condition). ◇

B EXPERIMENTAL RESULTS ON DECISION MODEL DISCOVERY FROM PROCESS MODELS

Industrial repository	Occurrence of patterns		
	P1	P2	P3
Health Insurance	93.87	18.40	23.31
IT Governance	71.43	42.86	35.71
Health Care	14.91	16.15	13.04
Energy Supplier A	67.70	19.47	7.96
Energy Supplier B	71.95	15.24	7.32
ISO	48.53	10.29	0.00
IT Service Management	61.44	22.22	3.27
Banking	100.00	66.67	33.33
Weighted average	63.26	17.83	10.04

Table 15: Occurrence frequency of the control-flow-based patterns in the industrial repositories

Industrial repository	Available	Refactored	
		Absolute	Relative
Health Insurance	163	155	95.09%
IT Governance	14	12	85.71%
Health Care	161	52	32.30%
Energy Supplier A	226	156	69.03%
Energy Supplier B	328	242	73.78%
ISO	68	33	48.53%
IT Service Management	153	96.00	62.75%
Banking	3.00	3.00	100.00%
Weighted Average			67.12%

Table 16: Amount of process models before and after the refactoring experiment in the industrial repositories

Industrial repository	Split XOR gateways		Activities	
	Before refactoring	After refactoring	Before refactoring	After refactoring
Health Insurance	4.48	4.05	13.59	13.66
IT Governance	4.57	2.08	11.67	12
Health Care	3.58	2.81	9.56	10.38
Energy Supplier A	5.03	4.26	16.96	17.15
Energy Supplier B	3.00	2.77	10.76	10.86
ISO	3.24	3.03	13.21	13.27
IT Service Management	4.57	3.76	15.57	15.69
Banking	3.00	2.00	3.00	3.33
Weighted Average	3.91	3.06	11.79	12.04

Table 17: Number of elements per model in industrial repositories before and after the refactoring experiment

Amount	Occurrence of patterns							
	$\Pi 1$	$\Pi 2$	$\Pi 3$	$\Pi 4$	$\Pi 5-a$	$\Pi 5-b$	$\Pi 6-a$	$\Pi 6-b$
Absolute	17	19	0	5	5	7	11	8
Relative	39.53 %	44.19 %	0.00 %	11.63 %	11.63 %	16.28 %	25.58 %	18.60 %

Table 18: Occurrence frequency of the data-centric decision patterns in the hospital repository from Section 8.2.3

C

EXPERIMENTAL RESULTS ON DECISION MODEL DISCOVERY FROM EVENT LOGS

Noise in %	Fold Number									
	1	2	3	4	5	6	7	8	9	10
0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
2	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
3	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
4	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
5	0.97	0.97	0.96	0.97	0.96	0.97	0.97	0.97	0.97	0.97
6	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
7	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
8	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
9	0.95	0.94	0.95	0.95	0.95	0.95	0.95	0.94	0.95	0.95
10	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94

Table 19: Mean accuracy of the discovered decision models in 10 folds for the synthetic loan assessment event log (cf. Section 8.3.2)

Noise in %	Fold Number									
	1	2	3	4	5	6	7	8	9	10
0	0.84	0.83	0.84	0.84	0.84	0.84	0.83	0.84	0.83	0.84
1	0.83	0.83	0.83	0.83	0.83	0.82	0.82	0.83	0.83	0.83
2	0.83	0.82	0.83	0.83	0.82	0.83	0.83	0.82	0.83	0.82
3	0.83	0.82	0.83	0.83	0.82	0.83	0.82	0.81	0.83	0.82
4	0.82	0.82	0.82	0.82	0.81	0.82	0.81	0.81	0.82	0.81
5	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.79	0.80	0.81
6	0.78	0.78	0.75	0.78	0.77	0.78	0.79	0.78	0.78	0.78
7	0.75	0.75	0.75	0.75	0.75	0.76	0.75	0.76	0.75	0.75
8	0.72	0.72	0.72	0.72	0.71	0.72	0.73	0.72	0.72	0.73
9	0.67	0.63	0.67	0.67	0.66	0.67	0.68	0.66	0.67	0.66
10	0.63	0.63	0.63	0.63	0.63	0.63	0.64	0.65	0.63	0.64

Table 20: Mean accuracy of the discovered decision models in 10 folds in the real-life roads fine event log (cf. Section 8.3.2)

Noise in %	Fold Number									
	1	2	3	4	5	6	7	8	9	10
0	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
1	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
2	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
3	0.98	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
4	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.98
5	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94
6	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94
7	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
8	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
9	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
10	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90

Table 21: Mean accuracy in 10 folds for the synthetic loan assessment of the decision models discovered from the credit-risk assessment event log by the NEFCLASS classifier (cf. Section 8.3.3)

Noise in %	Fold Number									
	1	2	3	4	5	6	7	8	9	10
0	0.96	0.98	0.92	0.99	0.95	0.97	0.96	0.96	0.99	0.99
1	0.99	0.95	0.94	0.92	0.93	0.94	0.96	0.96	0.98	0.98
2	0.95	0.97	0.96	0.91	0.95	0.97	0.96	0.95	0.98	0.96
3	0.93	0.89	0.97	0.88	0.96	0.91	0.89	0.89	0.93	0.94
4	0.95	0.95	0.91	0.91	0.93	0.95	0.89	0.91	0.94	0.92
5	0.93	0.88	0.93	0.93	0.87	0.95	0.92	0.92	0.92	0.92
6	0.93	0.92	0.83	0.85	0.92	0.96	0.87	0.90	0.89	0.86
7	0.92	0.92	0.87	0.82	0.88	0.94	0.94	0.88	0.86	0.86
8	0.85	0.88	0.92	0.87	0.84	0.84	0.81	0.85	0.86	0.8
9	0.85	0.78	0.82	0.84	0.73	0.81	0.73	0.81	0.74	0.84
10	0.83	0.74	0.87	0.73	0.82	0.86	0.85	0.91	0.87	0.78

Table 22: Mean accuracy in 10 folds for the synthetic loan assessment of the decision models discovered from the credit-risk assessment event log by the genetic classifier (cf. Section 8.3.3)

BIBLIOGRAPHY

[1] Promnicat. https://github.com/tobiashoppe/promnicat. [Online; accessed 27-July-2017].

[2] Camunda BPM Platform. https://www.camunda.org/. [Online; accessed 30-July-2017].

[3] CPN Tools. http://cpntools.org/. [Online; accessed 29-November-2016].

[4] CLIPS: A Tool for Building Expert Systems. http://clipsrules.sourceforge.net. [Online; accessed 14-May-2017].

[5] A Report: The EU General Data Protection Regulation . Allen & Overy LLP, 2016. URL http://www.allenovery.com/SiteCollectionDocuments/Radical%20changes%20to%20European%20data%20protection%20legislation.pdf.

[6] SAP Decision Service Management. http://scn.sap.com/docs/DOC-29158, 2016. [Online; accessed: 2017-02-14].

[7] Fabrizio Albertetti, Paul Cotofrei, Lionel Grossrieder, Olivier Ribaux, and Kilian Stoffel. The crilim methodology: Crime linkage with a fuzzy mcdm approach. In *Intelligence and Security Informatics Conference (EISIC), 2013 European*, pages 67–74. IEEE, 2013.

[8] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. A pattern language: Towns, buildings, construction (center for environmental structure). 1977.

[9] Isabelle Alvarez. Explaining the result of a decision tree to the end-user. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 411–415. IOS Press, 2004.

[10] Shuang An and Qinghua Hu. Fuzzy rough decision trees. In *International Conference on Rough Sets and Current Trends in Computing*, pages 397–404. Springer, 2012.

[11] Ken Arnold, James Gosling, and David Holmes. *The Java programming language*. Addison Wesley Professional, 2005.

[12] Bart Baesens, Rudy Setiono, Christophe Mues, and Jan Vanthienen. Using neural network rule extraction and decision tables for credit-risk evaluation. *Management science*, 49(3), 2003.

[13] Bart Baesens, Tony Van Gestel, Stijn Viaene, Maria Stepanova, Johan AK Suykens, and Jan Vanthienen. Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society*, 54(6):627–635, 2003.

[14] Marko Bajec and Marjan Krisper. A methodology and tool support for managing business rules in organisations. *Information Systems*, 30(6):423–443, 2005.

[15] Dennis Baker, Donald Bridges, Regina Hunter, Gregory Johnson, Joseph Krupa, James Murphy, and Ken Sorenson. Guidebook to decision-making methods. *Department of Energy, USA*, 2002.

[16] Paige Baltzan and Amy Phillips. *Business Driven Information Systems*. McGraw-Hill Higher Education, 2013.

[17] Kimon Batoulis, Anne Baumgrass, Nico Herzberg, and Mathias Weske. Enabling dynamic decision making in business processes with DMN. In *International Conference on Business Process Management*, volume 256 of *Lecture Notes in Business Information Processing*, pages 418–431, 2015.

[18] Kimon Batoulis, Andreas Meyer, Ekaterina Bazhenova, Gero Decker, and Mathias Weske. Extracting decision logic from process models. In *Proceedings of the 29th International Conference on Advanced Information Systems Engineering*, volume 9097 of *Lecture Notes in Computer Science*, pages 349–366. Springer International Publishing, 2015.

[19] Ekaterina Bazhenova. Support of decision tasks in business process management. In *Proceedings of the 2014 Central European Workshop on Services and their Composition*, volume 1140 of *ZEUS '14*, pages 21–26, 2014.

[20] Ekaterina Bazhenova and Mathias Weske. A data-centric approach for business process improvement based on decision theory. In *Proceedings of the 15th International Conference on Business Process Modeling, Development and Support*, volume 175 of *Lecture Notes in Business Information Processing*, pages 242–256. Springer Berlin Heidelberg, 2014.

[21] Ekaterina Bazhenova and Mathias Weske. Deriving decision models from process models by enhanced decision mining. In *Business Process Management Workshops*, volume 256 of *Lecture Notes in Business Information Processing*, pages 444–457. Springer International Publishing, 2015.

[22] Ekaterina Bazhenova and Mathias Weske. Optimal acquisition of input data for decision taking in business processes. In *Pro-*

ceedings of the Symposium on Applied Computing, pages 703–710. ACM, 2017.

[23] Ekaterina Bazhenova, Susanne Buelow, and Mathias Weske. Discovering decision models from event logs. In *In Proceedings of the 19th International Conference on Business Information Systems*, volume 255 of *Lecture Notes in Business Information Processing*, pages 237–251. Springer International Publishing, 2016.

[24] Ekaterina Bazhenova, Stephan Haarmann, Sven Ihde, Andreas Solti, and Mathias Weske. Discovery of fuzzy DMN decision models from event logs. In *Proceedings of the 29th International Conference on Advanced Information Systems Engineering*, volume 10253 of *Lecture Notes in Computer Science*, pages 629–647. Springer International Publishing, 2017.

[25] Ekaterina Bazhenova, Francesca Zerbato, and Mathias Weske. Data-centric extraction of DMN decision models from BPMN process models. In *Business Process Management Workshops*, volume 308 of *Lecture Notes in Business Information Processing*, pages 542–555. Springer International Publishing, 2018.

[26] Jörg Becker, Dominic Breuker, Patrick Delfmann, and Martin Matzner. Designing and implementing a framework for event-based predictive modelling of business processes. In *Enterprise modelling and information systems architectures*, volume 234 of *Lecture Notes in Informatics*, pages 71–84, 2014.

[27] Barnabás Bede. *Mathematics of Fuzzy Sets and Fuzzy Logic*, volume 295 of *Studies in Fuzziness and Soft Computing*. Springer, 2013. ISBN 978-3-642-35220-1.

[28] Luigi Benedicenti, Giancarlo Succi, Tulio Vernazza, and Andrea Valerio. Object oriented process modeling with fuzzy logic. In *Proceedings of the 1998 ACM symposium on Applied Computing*, pages 267–271. ACM, 1998.

[29] Pierre Berlandier, Eric Charpentier, and Duncan Clark. *Governing Operational Decisions in an Enterprise Scalable Way*. IBM Redbooks, 2013.

[30] Alexander Bock, Heiko Kattenstroth, and Sietse Overbeek. Towards a modeling method for supporting the management of organizational decision processes. In *Modellierung*, volume 225 of *Lecture Notes in Informatics*, pages 49–64. Gesellschaft für Informatik, 2014.

[31] Nicola Boffoli, Danilo Caivano, Daniela Castelluccia, and Giuseppe Visaggio. Business process lines and decision tables driving flexibility by selection. In *International Conference on Software Composition*, SC'12, pages 178–193. Springer, 2012.

[32] Carmen Bratosin, Natalia Sidorova, and Wil M.P. van Der Aalst. Distributed genetic process mining using sampling. In *International Conference on Parallel Computing Technologies*, volume 6873 of *Lecture Notes in Computer Science*, pages 224–237. Springer, 2011.

[33] Felix C. Brodbeck, Rudolf Kerschreiter, Andreas Mojzisch, and Stefan Schulz-Hardt. Group decision making under conditions of distributed knowledge: The information asymmetries model. *Academy of Management Review*, 32(2):459–479, 2007.

[34] Bertjan Broeksema, Thomas Baudel, Arthur G. Telea, and Paolo Crisafulli. Decision exploration lab: A visual analytics solution for decision management. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):1972–1981, 2013.

[35] Eric D. Browne, Michael Schrefl, and James R. Warren. Goal-focused self-modifying workflow in the healthcare domain. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, pages 10–25. IEEE, 2004.

[36] Canadian Standards Association. Z243.1-1970 for Decision Tables, 1970.

[37] Noel Capon. Credit scoring systems: A critical analysis. *Journal of Marketing*, 46(2), 1982.

[38] Gisele C. Cardoso and Fernando Antonio C. Gomide. Newspaper demand prediction and replacement model based on fuzzy clustering and rules. *Information Sciences*, 177(21):4799–4809, 2007.

[39] Filip Caron, Jan Vanthienen, and Bart Baesens. Business rule patterns and their application to process analytics. In *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2013 17th IEEE International*, pages 13–20. IEEE, 2013.

[40] Semra Catalkaya, David Knuplesch, Carolina Chiao, and Manfred Reichert. Enriching business process models with decision rules. In *BPM 2013 International Workshops*, volume 171 of *Lecture Notes in Business Information Processing*. Springer, 2013.

[41] Pei-Chann Chang, Chin-Yuan Fan, and Wei-Yuan Dzan. A cbr-based fuzzy decision tree approach for database classification. *Expert Systems with Applications*, 37(1):214–225, 2010.

[42] Gary Charness and Matthias Sutter. Groups Make Better Self-Interested Decisions. *Journal of Economic Perspectives*, 26(3):157–76, 2012.

[43] Michele Chinosi and Alberto Trombetta. Bpmn: An introduction to the standard. *Computer Standards and Interfaces*, 34(1): 124–134, 2012. ISSN 0920-5489.

[44] Stephen L. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2(3):267–278, 1994.

[45] Codasyl. *A Modern Appraisal of Decision Tables*. Report of The Decision Table Task Group, ACM, New York, 1982.

[46] Decision Management Community. https://dmcommunity.org/, 2017. [Online; accessed 27-July-2017].

[47] Paul Cotofrei and Kilian Stoffel. Fuzzy extended bpmn for modelling crime analysis processes. In *First International Symposium on Data-Driven Process Discovery and Analysis*, volume 116 of *Lecture Notes in Business Information Processing*, pages 13–28. Citeseer, 2011.

[48] Steve Craggs and Brian Safron. *Operational Decision Management For Dummies*. John Wiley and Sons, Inc., 2013. ISBN 978-1-118-67957-9.

[49] Brian J. Cragun and Harold J. Steudel. A decision-table-based processor for checking completeness and consistency in rule-based expert systems. *International Journal of Man-Machine Studies*, 26(5):633–648, 1987.

[50] Zhivka Dangarska, Kathrin Figl, and Jan Mendling. An explorative analysis of the notational characteristics of the decision model and notation (DMN). In *Workshop Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference*, pages 1–9. IEEE Computer Society, 2016.

[51] Thomas H. Davenport. *Process Innovation: Reengineering Work Through Information Technology*. Harvard Business School Press, Boston, MA, USA, 1993. ISBN 0-87584-366-2.

[52] Massimiliano De Leoni and Felix Mannhardt. Road traffic fine management process. https://data.4tu.nl/repository/uuid:270fd440-1057-4fb9-89a9-b699b47990f55, 2015. [Online; accessed 25-June-2017].

[53] Massimiliano de Leoni, Marlon Dumas, and Luciano García-Bañuelos. Discovering branching conditions from business process execution logs. In *Fundamental Approaches to Software Engineering*, volume 7793 of *Lecture Notes in Computer Science*, pages 114–129. Springer, 2013.

[54] Erwin De Ley and Dirk Jacobs. Rules-based analysis with jboss drools: adding intelligence to automation. *Proceedings of ICALEPCS 2011*, pages 790–793, 2011.

[55] A. K. Alves de Medeiros, A. J. M. M. (Ton) Weijters, and Wil M. P. van der Aalst. Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, 14(2):245–304, 2007.

[56] A.K. Alves De Medeiros and Christian W Günther. Process mining: Using CPN tools to create test logs for mining algorithms. In *Proceedings of the sixth workshop on the practical use of coloured Petri nets and CPN tools*, volume 576, 2005.

[57] A.K. Alves de Medeiros, Boudewijn F. Van Dongen, Wil M.P. van Der Aalst, and A.J.M.M (Ton) Weijters. Process mining: Extending the Alpha-algorithm to mine short loops. Technical report, BETA Working Paper Series (WP 113), Eindhoven University of Technology, Eindhoven, 2004.

[58] Johannes De Smedt, Faruk Hasić, Seppe K. L. M. vanden Broucke, and Jan Vanthienen. Towards a holistic discovery of decisions in process-aware information systems. In *International Conference on Business Process Management*, volume 10445 of *Lecture Notes in Computer Science*, pages 183–199. Springer International Publishing, 2017.

[59] Johannes De Smedt, Seppe K. L. M. vanden Broucke, Josue Obregon, Aekyung Kim, Jae-Yoon Jung, and Jan Vanthienen. Decision mining in a broader context: An overview of the current landscape and future directions. In *Business Process Management Workshops*, volume 281 of *Lecture Notes in Business Information Processing*, pages 197–207. Springer International Publishing, 2017.

[60] Tom Debevoise and James Taylor. *The MicroGuide to Process Modeling and Decision in BPMN/DMN*. CreateSpace Independent Publishing Platform, 2014.

[61] Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and analysis of business process models in BPMN. *Information and Software technology*, 50(12):1281–1294, 2008.

[62] Marlon Dumas, Wil M.P. van der Aalst, and Arthur H.M. Ter Hofstede. *Process-aware information systems: bridging people and software through process technology*. John Wiley & Sons, 2005.

[63] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management*. Springer, 2013.

[64] Reinhold Dunkl, Stefanie Rinderle-Ma, Wilfried Grossmann, and Karl Anton Fröschl. *A method for analyzing time series data in process mining: application and extension of decision point analysis*, volume 204 of *Lecture Notes in Business Information Processing*, pages 68–84. Springer International Publishing, 2015.

[65] Rami-Habib Eid-Sabbagh, Matthias Kunze, Andreas Meyer, and Mathias Weske. A platform for research on process model collections. *Business Process Model and Notation*, pages 8–22, 2012.

[66] Hillel J. Einhorn and Robin M. Hogarth. Behavioral decision theory: Processes of judgement and choice. *Journal of Accounting Research*, 19:1–31, 1981.

[67] Opher Etzion and Peter Niblett. *Event processing in action*. Manning Publications Co., 2010.

[68] Walid Fdhila, Conrad Indiono, Stefanie Rinderle-Ma, and Rudolf Vetschera. Finding collective decisions: Change negotiation in collaborative business processes. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, volume 9415 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2015.

[69] FICO. https://www.ficoanalyticcloud.com/, 2017. Accessed: 2017-02-14.

[70] Alan N. Fish. *Knowledge automation: how to implement decision management in business processes*, volume 595. John Wiley & Sons, 2012.

[71] Milton Friedman. *Capitalism and Freedom*. University of Chicago Press, 1962.

[72] Milton Friedman. *The Social Responsibility of Business Is to Increase Its Profits*, pages 173–178. Springer Berlin Heidelberg, 2007.

[73] Fabian Friedrich, Jan Mendling, and Frank Puhlmann. *Process Model Generation from Natural Language Text*, pages 482–496. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[74] Johny Ghattas, Pnina Soffer, and Mor Peleg. Improving business process decision making based on past experience. *Decision Support Systems*, 59(0):93 – 107, 2014.

[75] Stijn Goedertier and Jan Vanthienen. Business Rules for Compliant Business Process Models. In *Proceeding of the 9th International Conference on Business Information Systems (BIS 2006)*, volume P-85 of *Lecture Notes in Informatics*, pages 558–579. Gesellschaft fuer Informatik, 6 2006.

[76] Stijn Goedertier and Jan Vanthienen. Compliant and flexible business processes with business rules. In *7th Workshop on Business Process Modeling, Development and Support*, volume 236 of *CEUR Workshop Proceedings*, pages 94–104. CEUR-WS.org, 2006.

[77] Stijn Goedertier, Jan Vanthienen, and Filip Caron. Declarative business process modelling: Principles and modelling languages. *Enterprise Information Systems*, 9(2):161–185, February 2015.

[78] Herman Heine Goldstine, John Von Neumann, and John Von Neumann. *Planning and coding of problems for an electronic computing instrument*. Institute for Advanced Study Princeton, N. J, 1948.

[79] Antonio Gonzalez and Raul Perez. Completeness and consistency conditions for learning fuzzy rules. *Fuzzy Sets and Systems*, 96(1):37–51, 1998.

[80] Jaap Gordijn, Hans Akkermans, and Hans Van Vliet. Business modelling is not process modelling. In *International Conference on Conceptual Modeling*, volume 1921 of *Lecture Notes in Computer Science*, pages 40–51. Springer, 2000.

[81] Christian W. Günther and Wil M.P. van der Aalst. Fuzzy Mining – Adaptive Process Simplification Based on Multi-Perspective Metrics. In *Business Process Management*, volume 4714 of *Lecture Notes in Computer Science*, pages 328–343. Springer, 2007.

[82] Curt Hall. Enterprise decision management: Business intelligence advisory service. *Executive Report volume 5, Cutter Consortium*, 6, 2005.

[83] Michael Hammer and James A. Champy. *Reengineering the Corporation: A Manifesto for Business Revolution*. HarperBusiness, 1993.

[84] Faruk Hasic, Linus Vanwijck, and Jan Vanthienen. Integrating processes, cases, and decisions for knowledge-intensive process modelling. In *Proceedings of the 1st International Workshop on Practicing Open Enterprise Modeling within OMiLAB (PrOse 2017) co-located with 10th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modelling (PoEM 2027)*, volume 1999. CEUR-WS.org, 2017.

[85] Nico Herzberg, Andreas Meyer, and Mathias Weske. An event processing platform for business process management. In *Enterprise Distributed Object Computing*, pages 107–116. IEEE, 2013.

[86] Mark Hiscock, Guy Hindle, Mike Johnson, Tim Wuthenow, James Taylor, David Griffiths, and Graeme Everton. *Flexible Decision Management with Business Rules on IBM z Systems*. International Business Machines Corporation, 2015.

[87] Frank Hoffmann. Combining boosting and evolutionary algorithms for learning of fuzzy classification rules. *Fuzzy Sets and Systems*, 141(1):47–58, 2004.

[88] Frank Hoffmann, Bart Baesens, Jurgen Martens, Ferdi Put, and Jan Vanthienen. Comparing a genetic fuzzy and a neurofuzzy classifier for credit scoring. *International Journal of Intelligent Systems*, 17(11):1067–1083, 2002.

[89] Frank Hoffmann, Bart Baesens, C. Mues, T. Van Gestel, and J. Vanthienen. Inferring descriptive and approximate fuzzy rules for credit scoring using evolutionary algorithms. *European Journal of Operational Research*, 177(1):540–555, 2007.

[90] Edward M. Hubbard, Ilka Diester, Jessica F. Cantlon, Daniel Ansari, Filip van Opstal, and Vanessa Troiani. The evolution of numerical cognition: From number neurons to linguistic quantifiers. *The Journal of Neuroscience*, 28(46):11819–11824, 2009.

[91] Marta Indulska, Peter Green, Jan Recker, and Michael Rosemann. Business process modeling: Perceived benefits. In *International Conference on Conceptual Modeling*, volume 5829 of *lecture Notes in Computer Science*, pages 458–471. Springer, 2009.

[92] Marta Indulska, Michael zur Muehlen, and Jan Recker. Measuring method complexity: The case of the business process modeling notation. bpm center report bpm-09-03. *Business Process Management Center*, 2009.

[93] Donovan Isherwood and Marijke Coetzee. Trust cv: Reputation-based trust for collectivist digital business ecosystems. In *Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on*, pages 420–424. IEEE, 2014.

[94] Jan Vanthienen James Taylor, Alan Fish and Paul Vincent. *Emerging standards in decision modeling (BPM and Workflow Handbook Series)*. Future Strategies Inc., 2013.

[95] Monique H. Jansen-Vullers, P.A.M. (Ad) Kleingeld, and Mariska Netjes. Quantifying the performance of workflows. *Information Systems Management*, 25(4):332–343, 2008.

[96] Laurent Janssens, Ekaterina Bazhenova, Johannes De Smedt, Jan Vanthienen, and Marc Denecker. Consistent integration of decision {(DMN)} and process {(BPMN)} models. In *Proceedings of the 28th International Conference on Advanced Information*

Systems Engineering - Forum/Doctoral Consortium, volume 1612 of *CEUR Workshop Proceedings*, pages 121–128. CEUR-WS.org, 2016.

[97] Dierk Jugel, Christian M. Schweda, and Alfred Zimmermann. Modeling decisions for collaborative enterprise architecture engineering. In *Advanced Information Systems Engineering Workshops*, pages 351–362. Springer International Publishing, 2015.

[98] Kathrin Kirchner, Nico Herzberg, Andreas Rogge-Solti, and Mathias Weske. Embedding conformance checking in a process intelligence system in hospital environments. In *Process Support and Knowledge Representation in Health Care*, pages 126–139. Springer, Berlin, Heidelberg, 2013.

[99] Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. Systematic literature reviews in software engineering - a systematic literature review. *Information and Software Technology*, 51(1):7–15, January 2009. ISSN 0950-5849.

[100] George Klir and Bo Yuan. *Fuzzy sets and fuzzy logic*. Prentice hall New Jersey, 1995.

[101] Krzysztof Kluza and Krzysztof Honkisz. From SBVR to DMN Models. Proposal of Translation from Rules to Process and Decision Models. In *International Conference on Artificial Intelligence and Soft Computing*, volume 9693 of *Lecture Notes in Computer Science*, pages 453–462. Springer, 2016.

[102] Krzysztof Kluza, Krzysztof Kaczor, and Grzegorz J. Nalepa. Integration of business processes with visual decision modeling. presentation of the hades toolchain. In *Business Process Management Workshops*, volume 202 of *Lecture Notes in Business Information Processing*. Springer, 2014.

[103] Andreas Knöpfel, Bernhard Gröne, and Peter Tabeling. *Fundamental Modeling Concepts: Effective Communication of IT Systems*. Wiley, Chichester, UK, 2006.

[104] Elena Kornyshova and Rébecca Deneckère. Decision-making ontology for information system engineering. In *International Conference on Conceptual Modeling*, volume 6412 of *Lecture Notes in Computer Science*, pages 104–117. Springer, 2010.

[105] Marcello La Rosa, Marlon Dumas, Arthur H.M. Ter Hofstede, and Jan Mendling. Configurable multi-perspective business process models. *Information Systems*, 36(2):313–340, 2011.

[106] Robert K. Lai, Chin-Yuan Fan, Wei-Hsiu Huang, and Pei-Chann Chang. Evolving and clustering fuzzy decision tree for financial

time series data forecasting. *Expert Systems with Applications*, 36 (2):3761–3773, 2009.

[107] Geetika T. Lakshmanan, Davood Shamsi, Yurdaer N Doganata, Merve Unuvar, and Rania Khalaf. A markov prediction model for data-driven semi-structured business processes. *Knowledge and Information Systems*, 42(1):97–126, 2015.

[108] Sander J.J. Leemans, Dirk Fahland, and Wil M.P. van der Aalst. Discovering block-structured process models from incomplete event logs. In *International Conference on Applications and Theory of Petri Nets and Concurrency*, volume 8489 of *Lecture Notes in Computer Science*, pages 91–110. Springer, 2014.

[109] Kevin J. Leonard. The development of credit scoring quality measures for consumer credit applications. *International Journal of Quality and Reliability Management*, 12(4):79–85, 1995.

[110] Henrik Leopold, Jan Mendling, and Artem Polyvyanyy. Generating natural language texts from business process models. In *International Conference on Advanced Information Systems Engineering*, volume 7328 of *Lecture Notes in Computer Science*, pages 64–79. Springer, 2012.

[111] Henrik Leopold, Sergey Smirnov, and Jan Mendling. On the refactoring of activity labels in business process models. *Information Systems*, 37(5):443–459, 2012.

[112] Art Lew. Optimal conversion of extended-entry decision tables with general cost criteria. *Communications of the ACM*, 21(4): 269–279, 1978. ISSN 0001-0782.

[113] Jiafei Li, Dayou Liu, and Bo Yang. Process mining: Extending α-algorithm to mine duplicate tasks in process logs. In *Advances in Web and Network Technologies, and Information Management*, volume 4537 of *Lecture Notes in Computer Science*, pages 396–407. Springer, 2007.

[114] Heinz Lienhard and Urs-Martin Künzi. Workflow and business rules: a common approach. *Workflow Handbook*, pages 129–140, 2005.

[115] Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics*, volume 2 of *COLING '98*, pages 768–774. Association for Computational Linguistics, 1998.

[116] Brian C. Lovell and Christian J. Walder. Support vector machines for business applications. *Business Applications and Computational Intelligence*, page 267, 2006.

[117] Ruopeng Lu and Shazia Sadiq. A survey of comparative business process modeling approaches. In *International Conference on Business Information Systems*, volume 4439 of *Lecture Notes in Computer Science*, pages 82–94. Springer, 2007.

[118] Xixi Lu, Dirk Fahland, and Wil M.P. van der Aalst. Interactively exploring logs and mining models with clustering, filtering, and relabeling. In *BPM (Demos)*, volume 1789 of *CEUR Workshop Proceedings*, pages 44–49. CEUR-WS.org, 2016.

[119] David Luckham. The power of events: An introduction to complex event processing in distributed enterprise systems. In *International Workshop on Rules and Rule Markup Languages for the Semantic Web*, pages 3–3. Springer, 2008.

[120] Simone A. Ludwig, Domagoj Jakobovic, and Stjepan Picek. Analyzing gene expression data: Fuzzy decision tree algorithm applied to the classification of cancer data. In *2015 IEEE International Conference on Fuzzy Systems*, pages 1–8. IEEE, 2015.

[121] Rik Maes. On the representation of program structures by decision tables: A critical assessment. *The Computer Journal*, 21(4): 290–295, 1978.

[122] Felix Mannhardt, Massimiliano De Leoni, Hajo A. Reijers, and Wil M.P. van der Aalst. Decision mining revisited - discovering overlapping rules. volume 9694 of *Lecture Notes in Computer Science*, pages 377–392. Springer, 2016.

[123] Alberto Martelli and Ugo Montanari. Optimizing decision trees through heuristically guided search. *Communications of the ACM*, 21(12):1025–1039, dec 1978. ISSN 0001-0782.

[124] Raluca Meleancă. Will decision management systems revolutionize marketing? *Procedia - Social and Behavioral Sciences*, 92: 523 – 528, 2013. ISSN 1877-0428.

[125] Jan Mendling, Hajo A. Reijers, and Jan Recker. Activity labeling in process modeling: Empirical insights and recommendations. *Information Systems*, 35(4):467–482, 2010.

[126] Jan Mendling, Hajo A. Reijers, and Wil M.P. van der Aalst. Seven process modeling guidelines (7 PMG). *Information and Software Technology*, 52(2):127–136, 2010.

[127] Steven Mertens, Frederik Gailly, and Geert Poels. Enhancing declarative process models with dmn decision logic. In *International Conference on Enterprise, Business-Process and Information Systems Modeling*, volume 214 of *Lecture Notes in Business Information Processing*, pages 151–165. Springer, 2015.

[128] Andreas Meyer and Mathias Weske. Data Support in Process Model Abstraction. In *Conceptual Modeling*, pages 292–306. Springer, 2012.

[129] Andreas Meyer, Sergey Smirnov, and Mathias Weske. Data in business processes. *EMISA Forum*, 31(3):5–31, 2011.

[130] Andreas Meyer, Nico Herzberg, Frank Puhlmann, and Mathias Weske. Implementation Framework for Production Case Management: Modeling and Execution. In *Enterprise Distributed Object Computing (EDOC)*, pages 190–199. IEEE, 2014.

[131] Lev Muchnik, Sinan Aral, and Sean J. Taylor. Social influence bias: A randomized experiment. *Science*, 341(6146):647–651, 2013.

[132] Laura-Maria Murtola, Heljä Lundgrén-Laine, and Sanna Salanterä. Information systems in hospitals: a review article from a nursing management perspective. *IJNVO*, 13(1):81–100, 2013.

[133] Detlef Nauck. Fuzzy data analysis with nefclass. *International Journal of Approximate Reasoning*, 32(2):103 – 130, 2003.

[134] Mariska Netjes, Selma Limam Mansar, Hajo A. Reijers, and Wil M.P. van der Aalst. Performing business process redesign with best practices: an evolutionary approach. In *International Conference on Enterprise Information Systems*, volume 12 of *Lecture Notes in Business Information Processing*, pages 199–211. Springer, 2007.

[135] Tin A Nguyen, Walton Perkin, Thomas J Laffey, and Deanne Pecora. Knowledge base verification. *AI Magazine*, 8:69–75, 1987.

[136] OMG Group: Board of Directors. http://www.omg.org, 2017. [Online; accessed 27-July-2017].

[137] Board of Governors of the Federal Reserve System. Report to the Congress on Credit Scoring and Its Effects on the Availability and Affordability of Credit. Technical report, aug 2007. URL http://www.federalreserve.gov/boarddocs/RptCongress/creditscore/creditscore.pdf.

[138] OMG. Business Process Model and Notation (BPMN), v. 2.0.2, 2013.

[139] OMG. Business Motivation Model (BMM), v. 1.2, 2014.

[140] OMG. Unified Modeling Language (UML), v. 2.5, 2015.

[141] OMG. Case Management Model And Notation (CMMN), v. 1.1, 2016.

[142] OMG. Decision Model and Notation (DMN), v. 1.1, 2016.

[143] OMG. Semantics Of Business Vocabulary and Rules (SVBR), v. 1.4, 2017.

[144] Zdzisław Pawlak and Andrzej Skowron. Rough sets and boolean reasoning. *Information Sciences*, 177:41–73, 2006.

[145] Carla Marques Pereira and Pedro Sousa. A method to define an enterprise architecture using the zachman framework. In *Proceedings of the 19th ACM Symposium on Applied computing*, pages 1366–1371. ACM, 2004.

[146] Razvan Petrusel. Decision mining and modeling in a virtual collaborative decision environment. *Decision Support Systems: Advances In*, page 215, 2010.

[147] Razvan Petrusel. Using Markov decision process for recommendations based on aggregated decision data models. In *International Conference on Business Information Systems*, volume 157 of *Lecture Notes in Business Information Processing*, pages 125–137. Springer, 2013.

[148] Solomon L. Pollack, Harry T. Hicks, and William J. Harrison. *Decision tables: theory and practice*. Wiley-Interscience, New York, 1971. ISBN 0-471-69150-X.

[149] Jantima Polpinij, Aditya K. Ghose, and Hoa Khanh Dam. Business rules discovery from process design repositories. In *6th World Congress on Services*, pages 614–620. IEEE, 2010.

[150] Udo W. Pooch. Translation of decision tables. *ACM Computing Surveys*, 6(2):125–151, 1974.

[151] Michael E. Porter. *Competitive advantage: Creating and sustaining superior performance*. Free Press, 1985.

[152] John Winsor Pratt, Howard Raiffa, and Robert Schlaifer. *Introduction to statistical decision theory*. MIT Press, Cambridge, Mass. [u.a.], 1995.

[153] Maurizio Proietti and Fabrizio Smith. Reasoning on data-aware business processes with constraint logic. 14:60–75, 2014.

[154] Lila Rao, Gunjan Mansingh, and Kweku-Muata Osei-Bryson. Building ontology based knowledge maps to assist business process re-engineering. *Decision Support Systems*, 52(3):577–589, 2012.

[155] Manfred Reichert and Peter Dadam. Enabling adaptive process-aware information systems with adept2. In *Handbook of Research on Business Process Modeling*, pages 173–203. Information Science Reference, 2009.

[156] Manfred Reichert and Barbara Weber. *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*. Springer, 2012.

[157] Hajo A. Reijers and Selma Limam Mansar. Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. *Omega*, 33(4):283–306, 2005.

[158] Hajo A. Reijers, Selma Limam Mansar, and Wil M.P. van Der Aalst. Product-based workflow design. *Journal of Management Information Systems*, 20(1):229–262, 2003.

[159] Hajo A. Reijers, Sander van Wijk, Bela Mutschler, and Maarten Leurs. BPM in practice: Who is doing what? In *Business Process Management*, volume 6336 of *Lecture Notes in Computer Science*, pages 45–60. Springer, 2010. ISBN 978-3-642-15617-5.

[160] Willem De Roover and Jan Vanthienen. On the relation between decision structures, tables and processes. In *On the Move to Meaningful Internet Systems: OTM 2011 Workshops*, volume 7046 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2011.

[161] Daniela Rosca, Mark Feblowitz, and Chris Wild. A decision making methodology in support of the business rules lifecycle. In *Requirements Engineering*, pages 236–246. IEEE Computer Society, 1997.

[162] Michael Rosemann and Jan vom Brocke. The six core elements of business process management. In *Handbook on Business Process Management 1, Introduction, Methods, and Information Systems, 2nd Ed.*, pages 105–122. 2015.

[163] Arturo Rosenblueth and Norbert Wiener. The Role of Models in Science. *Philosophy of Science*, 12(4):316–321, 1945.

[164] Ali Asghar Anvary Rostamy, Meysam Shaverdi, Behnam Amiri, and Farideh Bakhshi Takanlou. Using fuzzy analytical hierarchy process to evaluate main dimensions of business process reengineering. *Journal of Applied Operational Research*, 4(2):69–77, 2012.

[165] Anne Rozinat and Wil M.P. van der Aalst. Decision mining in ProM. In *Proceedings of the 4th International Conference on Busi-*

ness Process Management, volume 4102 of *Lecture Notes in Computer Science*, pages 420–425. Springer International Publishing, 2006.

[166] Anne Rozinat and Wil M.P. van der Aalst. Decision mining in business processes. In *BPM Center Report BPM-06-10*. Beta, Research School for Operations Management and Logistics, 2006.

[167] Khodakaram Salimifard and Mike Wright. Petri net-based modelling of workflow systems: An overview. *European journal of operational research*, 134(3):664–676, 2001.

[168] SAS. http://www.sas.com, 2016. [Online; accessed: 2016-08-02].

[169] August-Wilhelm Scheer. *Business process engineering: Reference models for industrial enterprises*. Springer, 2 edition, 1994.

[170] August-Wilhelm Scheer, Oliver Thomas, and Otmar Adam. Process modeling using event-driven process chains. *Process-Aware Information Systems*, pages 119–146, 2005.

[171] Helen Schonenberg, Ronny Mans, Nick Russell, Nataliya Mulyar, and Wil M.P. van der Aalst. Process flexibility: A survey of contemporary approaches. In *CIAO! / EOMAS*, volume 10 of *Lecture Notes in Business Information Processing*, pages 16–30. Springer, 2008.

[172] Signavio. http://www.signavio.com, 2016. [Online; accessed: 2017-02-14.

[173] Bruce Silver. *BPMN Method and Style*. Cody-Cassidy Press, 2 edition, 2011.

[174] Herbert A. Simon and Allen Newell. Heuristic Problem Solving: The Next Advance in Operations Research. *Operations Research*, 6(1):1–10, 1958.

[175] Renuka Sindhgatta, Aditya Ghose, and Hoa Khanh Dam. Context-aware analysis of past process executions to aid resource allocation decisions. In *International Conference on Advanced Information Systems Engineering*, volume 9694 of *Lecture Notes in Computer Science*, pages 575–589. Springer, 2016.

[176] Jim Sinur. When rules go inside out with BPM. *Gartner*, 2007.

[177] James Taylor. *Decision Management Systems*. IBM Press, 1st edition, 2011. ISBN 0132884380, 9780132884389.

[178] James Taylor and Jan Purchase. *Real-World Decision Modeling with DMN*. Meghan-Kiffer Press, 2016. ISBN 0-929652-59-2.

[179] James Taylor and Neil Raden. *Smart Enough Systems: How to Deliver Competitive Advantage by Automating Hidden Decisions*. Pearson Education, 2007.

[180] Han van der Aa, Henrik Leopold, Kimon Batoulis, Mathias Weske, and Hajo A. Reijers. Integrated process and decision modeling for data-driven processes. In *Business Process Management Workshops*, volume 256 of *Lecture Notes in Business Information Processing*, pages 405–417. Springer International Publishing, 2015.

[181] Wil M.P. van der Aalst. Trends in business process analysis - from verification to process mining. In *ICEIS 2007 - Proceedings of the Ninth International Conference on Enterprise Information Systems*, pages 5–9, 2007.

[182] Wil M.P. van der Aalst. Process-aware information systems: Lessons to be learned from process mining. In *Transactions on petri nets and other models of concurrency II*, volume 5460 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2009.

[183] Wil M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer Publishing Company, Inc., 2011.

[184] Wil M.P. van der Aalst. *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.

[185] Wil M.P. van Der Aalst and Arthur H.M. Ter Hofstede. YAWL: yet another workflow language. *Information systems*, 30(4):245–275, 2005.

[186] Wil M.P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[187] Wil M.P. van der Aalst, Vladimir Rubin, H.M.W. (Eric) Verbeek, Boudewijn F. van Dongen, Ekkart Kindler, and Christian W Günther. Process mining: a two-step approach to balance between underfitting and overfitting. *Software & Systems Modeling*, 9(1):87, 2010.

[188] Boudewijn F. Van Dongen, Ana Karla A de Medeiros, H.M.W. (Eric) Verbeek, A.J.M.M (Ton) Weijters, and Wil M.P. van Der Aalst. The ProM framework: A new era in process mining tool support. In *ICATPN*, volume 3536, pages 444–454. Springer, 2005.

[189] Irene Vanderfeesten, Hajo A. Reijers, and Wil M.P. van der Aalst. Product based workflow support: dynamic workflow

execution. In *International Conference on Advanced Information Systems Engineering*, volume 5074 of *Lecture Notes in Computer Science*, pages 571–574. Springer, 2008.

[190] Jan Vanthienen. A note on English for decision tables considered harmful and the nested IF-THEN-ELSE. *SIGPLAN Notices*, 20(5):45–47, 1985.

[191] Jan Vanthienen and Frank Robben. Developing legal knowledge based systems using decision tables. In *Proceedings of the 4th International Conference on Artificial Intelligence and Law*, ICAIL '93, pages 282–291. ACM, 1993.

[192] Jan Vanthienen, Elke Dries, and Jeroen Keppens. Clustering knowledge in tabular knowledge bases. In *Eigth International Conference on Tools with Artificial Intelligence*, pages 88–95, 1996.

[193] Jan Vanthienen, Geert Wets, and Guoqing Chen. Incorporating fuzziness in the classical decision table formalism. *International journal of intelligent systems*, 11(11):879–891, 1996.

[194] Jan Vanthienen, Christophe Mues, and Stijn Goedertier. Experiences with modeling and verification of regulations. In *Proceedings of an International Workshop on Regulations Modelling and their Validation and Verification*, volume 241 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[195] H.M.W. (Eric) Verbeek, Joos C.A.M. Buijs, Boudewijn F. Van Dongen, and Wil M.P. van der Aalst. XES, XESame, and ProM 6. In *Forum at the Conference on Advanced Information Systems Engineering*, volume 72 of *Lecture Notes in Business Information Processing*, pages 60–75. Springer, 2010.

[196] Maurice Verhelst. *De Praktijk van Beslissingstabellen*. Kluwer, Deventer/Antwerpen, 1980.

[197] Barbara Von Halle and Larry Goldberg. *The Decision Model*. Taylor and Francis Group, 2010.

[198] Karen Walzer, Tino Breddin, and Matthias Groch. Relative temporal constraints in the Rete algorithm for complex event detection. In *Proceedings of the second international conference on distributed event-based systems*, pages 147–155. ACM, 2008.

[199] Wei Wang, Marta Indulska, and Shazia Sadiq. *To Integrate or Not to Integrate – The Business Rules Question*, volume 9694 of *Lecture Notes in Computer Science*, pages 51–66. Springer International Publishing, 2016.

[200] Hans Weigand, Willem-Jan van den Heuvel, and Marcel Hiel. Rule-based service composition and service-oriented business

rule management. In *Proceedings of the International Workshop on Regulations Modelling and Deployment*, pages 1–12. Citeseer, 2008.

[201] A.J.M.M (Ton) Weijters, Wil M.P. van der Aalst, and A.K. Alves De Medeiros. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Technical Report*, 166:1–34, 2006.

[202] Mathias Weske. *Business Process Management - Concepts, Languages, Architectures, 2nd Edition*. Springer, 2012.

[203] Branimir Wetzstein, Asli Zengin, Raman Kazhamiakin, Annapaola Marconi, Marco Pistore, Dimka Karastoyanova, and Frank Leymann. Preventing KPI violations in business processes based on decision tree learning and proactive runtime adaptation. *Journal of Systems Integration*, 3(1):3–18, 2012.

[204] Roel J. Wieringa. *Design science methodology for information systems and software engineering*. Springer Verlag, 2014.

[205] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. ISBN 0120884070.

[206] Petia Wohed, Wil M.P. van der Aalst, Marlon Dumas, Arthur H.M. ter Hofstede, and Nick Russell. On the Suitability of BPMN for Business Process Modelling. In *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 161–176. Springer, 2006.

[207] Anita Williams Woolley, Christopher F Chabris, Alex Pentland, Nada Hashmi, and Thomas W Malone. Evidence for a collective intelligence factor in the performance of human groups. *Science*, 330(6004):686–688, 2010.

[208] Feng Wu, Laura Priscilla, Mingji Gao, Filip Caron, Willem De Roover, and Jan Vanthienen. Modeling decision structures and dependencies. In *OTM Workshops*, volume 7567 of *Lecture Notes in Computer Science*. Springer, 2012.

[209] Lofti A. Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[210] Alireza Zarghami, Brahmananda Sapkota, Mohammad Zarifi Eslami, and Marten van Sinderen. Decision as a service: Separating decision-making from application process logic. In *Proceedings of the 16th International Enterprise Distributed Object Computing Conference*, pages 103–112. IEEE, 2012.

[211] Hans-Jürgen Zimmermann. *Fuzzy Set Theory and Its Applications (3rd Ed.)*. Kluwer Academic Publishers, 1996.

[212] Michael Zur Muehlen and Jan Recker. How much language is enough? Theoretical and practical use of the business process modeling notation. In *Seminal Contributions to Information Systems Engineering*, volume 5074 of *Lecture Notes in Computer Science*, pages 429–443. Springer, 2013.