WWU
MÜNSTER

Fundamental Data Mining Techniques for Declarative Process Mining

Nico Grohmann

OLMS

Fundamental Data Mining Techniques for Declarative Process Mining

Inauguraldissertation zur Erlangung des akademischen Grades eines Doktors der Wirtschaftswissenschaften durch die Wirtschaftswissenschaftliche Fakultät der Westfälischen Wilhelms-Universität Münster

Vorgelegt von

Nico Grohmann, MSc
aus Gießen

Dezember 2021

Dekan	Prof. Dr. Gottfried Vossen
Erster Gutachter	Prof. Dr. Gottfried Vossen
Zweiter Gutachter	Prof. Dr. Dr. h.c. Dr. h.c. Jörg Becker
Beisitzerin	Prof. Dr. Sonja Gensler
Mündliche Prüfung	11.01.2022

Nico Grohmann

Fundamental Data Mining Techniques for Declarative Process Mining

WWU
MÜNSTER

Wissenschaftliche Schriften der WWU Münster

Reihe IV

Band 21

Nico Grohmann

Fundamental Data Mining Techniques for Declarative Process Mining

Georg Olms Verlag
Hildesheim · Zürich · New York

Wissenschaftliche Schriften der WWU Münster
herausgegeben von der Universitäts- und Landesbibliothek Münster
http://www.ulb.uni-muenster.de

Eine Publikation in Zusammenarbeit mit dem Georg Olms Verlag
https://www.olms.de

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über https://www.dnb.de abrufbar.

Dieses Buch steht gleichzeitig in einer elektronischen Version über den Publikations- und Archivierungsserver der WWU Münster zur Verfügung.
https://www.ulb.uni-muenster.de/wissenschaftliche-schriften

Nico Grohmann
„Fundamental Data Mining Techniques for Declarative Process Mining“
Wissenschaftliche Schriften der WWU Münster, Reihe IV, Band 21
Georg Olms Verlag, Hildesheim

Zugl.: Diss. Universität Münster, 2022

ISBN 978-3-487-16155-6 (Druckausgabe Georg Olms Verlag)
ISBN 978-3-8405-0266-8 (elektronische Version)
DOI 10.17879/54059509128 (elektronische Version)
URN urn:nbn:de:hbz:6-54059510537 (elektronische Version)

direkt zur Online-Version:

Satz: Nico Grohmann
Titelbild: Nico Grohmann (WordCloud)
Umschlag: ULB Münster

Preface

Process mining has been a hot topic for the last 10 years. It originated from the work of Wil van der Aalst and has quickly picked up attention in both industry and academia. Indeed, industry has developed so much interest in the topic, its applications, and its potential benefits that several start-ups focusing on process mining could successfully established, one of which has even reached unicorn status.

Process mining is based on the idea of generating process models from event logs. In other words, it is a total deviation from process modeling, which has been popular for the last 25 years, since the interest is no longer to model a process before it is enacted and deployed, but afterwards. An event log as produced by almost every modern IT system is a protocol of all events which have occurred in a system up to a certain point in time; hence it shows both what has been executed (which software) and the order in which execution events have occurred. Obviously, process mining thus cannot capture many details which could easily be reflected in a process model (e.g., the human actors, roles or re-sources involved in the execution of a process, the data model underlying the events which are executed, or a relevant risk or SWOT model). On the other hand, various applications seem satisfied with the results that process mining can produce.

Several users as well as researchers see process mining as a special form of data mining. This analogy is valid, since an event log can be considered as a dataset from which a model (the process model) is learned, and then

compared to the intention or purpose that is underling the execution of the respective process (and for which a model may or may not yet exist). Due to this analogy, it makes sense to study whether and which established data mining techniques can be used in this context.

Nico Grohmann's thesis has two main research questions it wants to answer: How to develop an implementation which applies association rule and sequential pattern mining to event data in the form of event logs for use in practice, and how can resulting rules and patterns be translated into declarative model elements in a way that they can be exploited for process improvement. The reader's attention is particularly directed to Chapter 5, one of the main contributions, where he applies pattern and rule mining to event log data, and shows in detail how algorithms Apriori/FP-Growth and GSP can be applied by using the commercial RapidMiner tool. It is also shown how the resulting rules or sequential patterns can be translated into Declare statements. He then evaluates the approach on three vastly realistic examples, and delivers the insight that sequential mining can yield considerably better results than just rule mining.

The core of the thesis is a demonstration that process mining in the declarative direction can more or less easily be done using a commercially available tool without extraordinary efforts, which makes it an interesting read especially for the practitioner.

Münster, March 2022 Prof. Dr. Gottfried Vossen

Acknowledgements

I would like to thank my doctoral supervisor Prof. Dr. Gottfried Vossen for his supervision and assistance throughout the research process that resulted in my thesis and the possibility to come to Münster to work as a research assistant in his group. He encouraged me to search for research opportunities in the context of process mining, which I consider a very promising technology for present and future Business Process Management activities. Both academically and personally, I developed myself significantly during the time under his supervision. Furthermore, I would like to extend my gratitude to Prof. Dr. Jörg Becker for his willingness to serve as the second reviewer, and Prof. Dr. Sonja Gensler for acting as the third examiner in the defense committee.

The relationship with my colleagues strongly influenced my time at the DBIS group. With no exception, I only made positive experiences while collaborating with them. In particular, I would like to thank Dr. Denis Martins and Dr. Felix Nolte for academic and non-academic talks in the office, and for reviewing parts of my thesis. In sum, I could benefit from them a lot by watching their journey to successful graduation. The same gratitude goes to Julia Seither for her administrative support and helpful advice. More than once, she provided me with assistance for tasks that were not necessarily her responsibility. Furthermore, I would like to thank Dr. Jens Lechtenbörger for his experienced guidance through the procedures of the university, Dr. Leschek Homann for the nice colleagueship including helpful talks about the doctorate, Jan Everding for the joint

work and being a pleasant office mate during the time we shared an office, Ralf Farke for his competent technical support, and Raquel Mello for the interesting conversations during my time at the institute.

Finally, I would like to thank my family, in particular my parents Dr. Walter and Karin Grohmann, my brother Marco Grohmann, and grandparents Walter and Erika Grohmann and Hans Isert who supported me during my doctorate with the confidence and optimism to eventually bring it to a successful conclusion. Special thanks go to my parents who enabled me to study which made the doctorate possible in the first place.

Münster, March 2022 Nico Grohmann

Contents

List of Figures

List of Tables

1 Introduction

Digitalization has influenced nearly all areas of life and is an unstoppable phenomenon for upcoming times. Whether it is in the communication, shopping, entertainment, or mobility area, digitalization has changed business models and how these actions are carried out. Former paper-based processes are now possible only using a digital device or have become entirely obsolete. Furthermore, new business models and processes were just not possible before digitalization had entered the stage [1].

Digitalization comprises a multi-dimensional concept whose capture is not trivial. For this purpose, research has developed various digital maturity models [TMB20] that try to capture the current digital status of an organization, whereby business processes and data were discovered to be of prime importance.

Processes characterize the creation of value for any business. Traditionally, Business Process Management (BPM) is the field that deals with all aspects of processes in organizations. It focuses on understanding and modeling existing processes and identifying the differences between desired process executions. BPM has its origins early in the 1990s, with the work of Hammer [Ham90] introducing Business Process Reeingeneering (BPR), claiming that companies should rather completely rework their work procedures instead of automating them. Today, processes are often supported by at least one information system and, therefore, produce large amounts of data [MSW11]. Processes consume data to fulfill their purposes and produce new data based on their executions. Exploiting

data sources is crucial for a large majority of business models today. The data mining discipline, nowadays often associated with the *Big Data* term, focuses on exploiting all kinds of data sources to understand or predict relationships in a domain [TSK16].

Recently, data mining techniques have been applied to process data in the form of so-called event logs. Process mining [Wil16] has arisen, which uses the traces of process executions in IT systems to discover real-world processes. With that technology, process modeling is now a semi-automated task that does not require manual modeling activities. In general, process mining connects the BPM and data mining disciplines by exploiting data sources for all kinds of process analyses.

Research progress in process mining directly contributes to advancements in BPM which can be seen as one dimension of digitalization. Like other research efforts in the information systems discipline, process mining is closely linked between research developments and the technology application in practice. New findings in research are applied in practice to verify whether they hold in reality. Reversely, observations in the industry lead to research initiatives to explain observations or come up with solutions for a problem observed. As process mining is a relatively new technology and its application in practice is only at the beginning in many environments, questions arise of how to integrate it in the existing BPM environment and which benefit it can deliver in what kind of circumstances.

1.1 Motivation

Process Mining allows companies and other organizations to get an overview of the actual status of their business processes. The foundations for process mining were laid in 2012 with the Process Mining Manifesto [vdAAdM+12] with van der Aalst being one of its principal founders

who published a work that summarizes a large share of the knowledge about this technology in 2016 [Wil16]. As a subset of BPM, process modeling tries to capture and visualize the flow of processes happening in any organization. Such activity typically results in a process model or diagram in a notation like Petri nets or BPMN. Typically, process mining applications in larger companies consider Order-to-Cash (O2C), Purchase-to-Pay (P2P), production or administrative processes. Such processes are often supported by various information systems, whereby some may even communicate with external systems from customers or suppliers [24]. In many cases, modeling such processes is not feasible [DGDMM15] because, depending on the abstraction degree, they consist of many activities and have many variants, meaning that the activities can be gone through in various ways. There are no or only a few conditions and dependencies that enforce a particular order of the process. The flow of the process is relatively free as long as it adheres to the logical structures (e.g., XOR-gateways) and starts in a start and ends in an end state.

For this reason, process mining, where the process model is automatically generated based on the input of process execution data, may be a suitable option. Van der Aalst et al. define process mining as a technique "to discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs readily available in today's (information) systems" [vdAAdM$^+$12]. The challenge is to collect all relevant event data from the various information systems and integrate and transform them into one event log. Assuming this hurdle is cleared, the file can be imported into a process mining tool of choice. Then, a process discovery routine generates a process model underlying the event data input. In some cases, this results in a large, complex, spaghetti-like diagram that is confusing and does not deliver any concise overview.

Still, modelers and other process stakeholders would like to know the process's most critical procedures and relations. Declarative model-

ing [DGDMM15] notations can help here as they introduce constraints and conditions on the process without defining the exact flow of the process in detail. Such constraints and conditions can originate from either process mining activities taking the event log as the source or manual modeling activities by process modelers using their domain knowledge about the process or conducting interviews with employees who are part of the process under consideration. In this way, even though traditional process modeling and mining cannot deliver satisfying analysis results, process analysts could nevertheless deliver possibly meaningful insights that help to improve the process performance.

All process stakeholders, including management people, must understand the resulting process models to ensure the analyses are beneficial. One widely-used notation for modeling processes in a declarative manner is Declare [PSVdA07]. It consists of a set of constraint templates restricting the existence of single activities or the relation of two or more activities between each other. Besides, a multi-perspective extension of Declare can also capture data-related conditions for their associated activities. As declarative notations for process modeling are nowhere near as widely used as imperative notations like the industry-standard Business Process Model and Notation (BPMN), employees that are not professional process modelers may have difficulties grasping the process depicted in such models or at least need some time to familiarize themselves [FLM+09]. Therefore, another format with which non-experts in modeling are more familiar but can still represent the process coherences may be desirable.

For a long time, data mining has used association rules [ZB03] and sequential patterns [16] to describe relations in all kinds of datasets, whereby the market basket analysis is the most prominent application scenario. Association rules create a statement that whenever a customer buys product A, he or she is likely to buy product B, whereas sequential patterns show frequent sequential orders in which customers buy products. Because

of their conciseness, both association rules and sequential patterns are intuitively understandable for most employees with various backgrounds. Transferring their application from market basket to process analysis means that the activities of a process are the new products and a process instance is one customer transaction describing the products a customer buys. In this way, they are an intermediary representation of process relations but not yet translated to model elements of the declarative process model notation. Ideally, they are derived directly from the event log input that is the same as for "traditional" process mining resulting in process graphs.

Consider the following motivating example that may happen similarly in practice based on a sample event log from the internet. It was used for the BPI Challenge 2019 [18] and describes the purchase order process of a larger multi-national company based in the Netherlands in one year. While the scenario is fictive, the goal is to illustrate the benefits of combining the declarative process modeling paradigm and process mining in such application settings.

Motivating Example

The setting is a company *Paint & Co.* producing coatings and paints for industrial applications. Controller C detects a notification at her Business Intelligence (BI) dashboard indicating that recently revenue numbers have dropped. After a chat with her colleagues in the production department, she finds out that production had to reduce the production speed as the company lacks raw material. Furthermore, she hears that supply companies complained about the unreliability of the purchasing department of *Paint & Co.* C wants to get to the bottom of things and talks to a process analysis department colleague.

The purchase order process is currently not part of the department's analyses as it was found to be very hard to be modeled because of its

large number of variants. Furthermore, the focus of the process analysis team was set on the production processes as higher cost-saving and performance improvement potentials were expected there. The process analysis team recently included process mining into their skills portfolio and toolbox as the team thinks it could benefit the company substantially. However, the application of process mining in *Paint & Co.* is still at the beginning. There have been successful pilot projects with more straightforward and smaller processes that have led to changes and optimizations. One reason why process mining often has not yet been established for larger, more complex processes is that data collection and integration requires significant effort with the introduction of ETL (Extract, Transform, Load) pipelines. C and her colleague agree that they should apply process mining on the purchase order process to clarify the reason for the shortage of raw material.

They contact the purchasing department of *Paint & Co.* and explain their project. The purchasing department provides them with an extract of all ERP system tables that come in contact with the purchase order process. With these extracts, BI and process analysts, with the help of a data science expert, build an event log that describes the executions of the purchase order process in the last five years. After importing the file in a process mining tool (Disco [4] in this example), they start the process discovery process. It results in the process model shown in Figure 1.1.

C and her colleague from the process analysis team are surprised because the model is confusing even though it only includes 50 percent of the paths. They can identify the happy path of the process through the color coding and varying thickness of the output, indicating which of the activities and paths are frequent. First, *Paint & Co.* creates a purchase order item (activity *Create Purchase Order Item*) which includes the products the company would like to buy and leads to the buying contract between the company and the suppliers. Then, the vendor creates an invoice, and

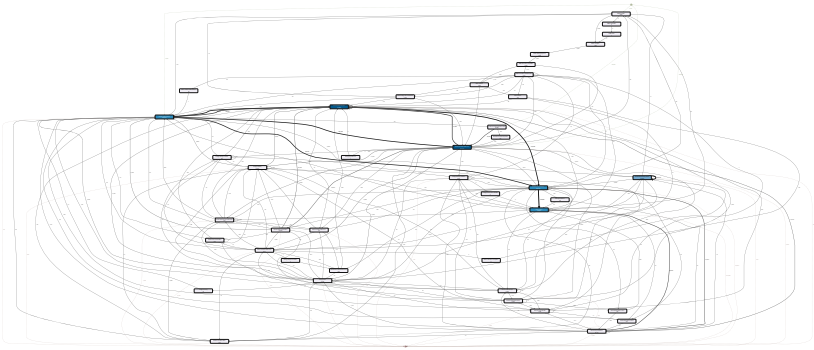


Figure 1.1: Discovery of Purchase Order Process in Disco [4].

the goods and invoice receipts are recorded, i.e., stored in the system. In some cases, the invoice creation is skipped or executed after the receipt of the goods has been recorded. Finally, the invoice is closed and archived (activity *Clear Invoice*). C and her colleague do not identify any significant deviations from what is expected and cannot find the problem that leads to the missing raw materials as relevant statistics like throughput time are in the norm.

Therefore, they decide to split the event data and only load the purchase order process cases of the last two months into the process mining tools because this is roughly the time where the complaints of the suppliers and the lack of raw material started. After generating the process model, it can be seen that the frequency of activities in the cases has changed. The happy path remains unchanged, but another path from *Vendor creates invoice* to *Create Purchase Requisition Item* back to the *Create Purchase Order Item* activity grows substantially regarding frequency. In a significant amount of cases, *Paint & Co.* requests new items after the vendor has already created the invoice.

Such process behavior is unusual and represents an exceptional case that requires further investigation. Now that C and her colleague have an indication of why the purchase order process may currently not be optimal and cause the problems with the procurement of the raw materials, they still do not know why the purchasing department executes the purchase orders in this way. A small interview session with the employees does not lead to new conclusions because they state that nothing has changed and the purchase orders are executed in the same way like they have been in the previous years. The BI and process analyst conferred to discuss the next steps with this unsatisfying situation.

The process analyst proposes to apply association rule and sequential pattern mining on the event log to derive frequent relations on the process. Thereby, these fundamental data mining techniques shall be enriched with data, i.e., attribute values associated with each activity execution. In this way, the execution path of the process is linked with attribute values. Still, it requires some domain knowledge to interpret the resulting rules and patterns and decide whether they represent behavior outside the happy path that should generally be avoided or one should try to keep infrequent. The data science of team of *Paint & Co.* supports the process of applying the two data mining techniques on the event log and develops a Python script that produces association rules and sequential patterns based on minimum support and confidence values. Again, the script uses the data from the last two months. After playing around with these settings for some time, C and her colleague come across the following sequential pattern, with relatively high support of 0.5, raising their interest:

{Vendor Creates Invoice, Create Purchase Requisition Item/user_005, Create Purchase Order Item}

The *user_005* value originates from the *User* attribute of the event log, which is a so-called event attribute that specifies the user who was

responsible for executing a particular activity (if known and meaningful, can also be empty). A slash between *Create Purchase Requisition Item* and *user_005* indicates that this specific value for the *User* attribute and the activity appeared together. The pattern shows that the rework of the purchase order items is significantly associated with one particular user. C and her colleague arrange a confidential talk with the head of the procurement department. The de-anonymization of the *user id* shows that the employee associated is relatively new in the company. Here, it is crucial always to be cautious of sensitive user-related that could lead to accusations of individual employees. Privacy issues are an important (research) topic for the process mining field.

The problem solution is found after a private face-to-face talk of the procurement manager and the employee, including a live walkthrough through the purchase order process. Instead of creating new purchase orders for every product *Paint & Co.* wants to get delivered, the employee created a purchase requisition item based on the same invoice number when he believed that the additional products have a content-related connection to an already existing order. In this way, the supply company receives a purchase order associated with an invoice number that already exists and that was possibly already closed and archived, interrupting the automatic workflow at the supplier's side. An employee of the sales department has to manually inspect the case and initiate the shipment of the products, causing delivery delays. This explains why *Paint & Co.* lacks some essential raw materials as the company applies Just-in-time production with minimal storage capacity from which it could draw on.

Still, the possibility to create a purchase requisition item based on an existing invoice is intended and was designed for cases where the supply company could not deliver parts of the order. In some of these cases, *Paint & Co.* falls back for other suppliers to obtain the products. However, a supplier may generally be the only one selling one type of material or

others do not have it in stock as well. The purchase requisition shall explicitly state that *Paint & Co.* asks the supplier to restock these products to deliver them in the future even though this could take longer than usual and therefore violate delivery time agreements. In total, the supply company could also use the findings of the sequential pattern analysis to make their sales processes more robust against such deviations so that they do not lead to the loss of automation.

C and her colleague are content with the outcome of their joint project. They could find the cause of the short raw materials. For the future, they agree that applying association rule and sequential pattern mining to process event data seems promising and should be continued with other processes, especially those that were not touched before by process analysis due to their complexity and deep integration in various company parts. The association rules and sequential patterns were easily understandable, also by non-process-experts, whereby sequential patterns were found to be more beneficial for understanding the process steps chronology.

Some questions remain open, for instance, how this new procedure of applying fundamental data mining techniques on event data should be included in the current process analysis cycle. Currently, the process analysis team of *Paint & Co.* starts with context analysis of the process and conducts interviews with employees who work with the process afterwards, which results in process models. Should process mining, in general, happen after a context analysis, and should, if possible, both modeling and mining be performed for one process? In case the process analysis team performs both techniques, how should *Paint & Co.* deal with non-matching results? Related to that is how to represent the association rules and sequential patterns in a global process model.

It is already clear that a declarative modeling notation with the opportunity to model data-related relations fits this purpose best here. Still, there is the question of which rules and patterns lead to which model

elements in the declarative modeling notation. C and her colleague raise the question of whether the notations currently described in the literature are sufficient or they have to be extended to make them more fitting to represent the mining output. Furthermore, both agree that general guidelines providing indications about parameter settings and the question of which rules and patterns should be selected for including them in a declarative model could be immensely helpful.

1.2 Research Objectives

The research foci of this thesis are two-fold. One part concentrates on rather technical issues of the implementation. It shows how a general implementation of association rule and sequential pattern mining could look like in programming languages dealing with datasets. Thereby, the thesis expands on all preprocessing steps and parameter settings in detail to finally apply the two fundamental data mining techniques. Additionally, it illustrates how various types of resulting association rules and patterns can be transformed to a Declare constraint to be included in a declarative process model based on a compact sample process event data set. The following questions capture the main contributions of this thesis.

- **RQ1** How to develop a general implementation of applying association rule and sequential pattern mining to event data in the form of event logs for use in practice?
- **RQ2** How can resulting rules and patterns be translated to declarative model elements, and are the existing (multi-perspective) notations appropriate and fitting to capture their meaning?

Another aspect focuses on integrating the declarative process mining approach presented in this thesis and the declarative paradigm in general

into a company's structure. It addresses how and for whom a rule/pattern and declarative model representation may be beneficial, compared to existing declarative process mining approaches whose algorithmic details may not be transparent to non-technical users. Both parts are supported by applying the approach to sample event logs from web repositories; however, real-world applications are not included and left for future research interests. Therefore, the following two questions are not explicitly answered but touched in the course of the thesis.

- Is the association rule and sequential pattern representation beneficial for users, especially those without knowledge about algorithmic details and programming languages?
- How should the application of association rule and sequential pattern mining be integrated into the BPM life cycle to maximize a company's benefit for process understanding?

1.3 Thesis Structure

The remaining thesis is structured as follows: Part 1 encompasses the foundations that play a substantial role throughout the thesis. Chapter 2 introduces a maturity model to assess a company's degree of digitalization concerning a set of dimensions. It identifies processes as the most critical aspect of digitalization and motivates the joint consideration of Business Process Management (BPM) and process mining. Both concepts or research fields are introduced in more detail in Chapter 3. Process mining enriches the traditional BPM field with data-driven analyses of real-world processes. Together, their application could result in findings for future process analysis initiatives that were not possible before. Next, Chapter 4 recalls association rule and sequential pattern mining as two fundamental data mining techniques.

Part 2 of this thesis combines the data and process mining techniques. Starting from their traditional, typical application settings like market basket analysis, Chapter 5 transfers them to the process analysis field and sets up a process to apply them to event logs for process mining. Furthermore, the chapter shows how the resulting rules and patterns can be translated to model elements in the (multi-perspective) Declare notation and how these notations can be adapted to represent them even more precisely. Chapter 6 evaluates the approach by applying it to sample event logs of different sizes from web repositories. This results in full declarative models and assesses whether and how process analysis and understanding benefit from this representation.

At last, Part 3 relates the contribution of this thesis with the research context and compares it with existing declarative process mining approaches. The goal is to demonstrate that the straightforward generation of association rules and sequential from event logs benefits the common understanding of a process model and a transformation to declarative model elements and backwards is practical. Finally, Chapter 8 concludes the thesis and points out possible directions of future research.

Part I

Foundations

Part 1 (“Foundations”) introduces the fundamental terms and concepts relevant to this thesis. Starting from a brief overview of digitalization or digital transformation of companies, Chapter 2 focuses on the digital dimension of *Processes*. Processes in the firm context are very strongly related to Business Process Management (BPM), which covers all aspects of business processes in organizations and represents a major research field. In that context, the Horus method represents a comprehensive framework guiding companies to analyze all aspects related to business processes and their connection to other business parts. It shall serve as a reference point to show how the approach presented in this thesis can be integrated into a company’s BPM activities.

Chapter 3 continues with process mining as a relatively new technology that exploits traces of processes supported by information systems of all kinds. Furthermore, it provides a short history of process mining and explains the fundamentals and most important types to use in practice. Both process modeling and mining are traditionally applied by using an imperative paradigm. However, researchers have started to investigate the declarative paradigm. Declarative approaches for process modeling do not define a control flow but describe the process at hand using constraints and conditions. In that way, the process is restricted by constraints. Every other behavior not captured by these constraints is allowed. Declarative model elements are especially useful for rather complex and loosely-structured processes. Their description with traditional process models would result in so-called "spaghetti" diagrams. This chapter introduces declarative constraints for process modeling, specifically the Declare notation, which is common and accepted in research environments. Then, a multi-perspective extension of Declare (MP-Declare) is presented which makes it possible to add attributes or other data-based conditions to the declarative process model. Examples of such conditions are activation, correlation, and time conditions on or between the process activities.

After setting the basis for the process field, Chapter 4 continues with introducing two fundamental data mining techniques, namely association rule mining and sequential pattern mining. Starting with the Apriori principle, the standard market basket example recalls the advantages of the FP-Growth algorithm for association rule and the GSP algorithm for sequential pattern mining. The overall goal is to use these techniques on event data (in the form of event logs) to discover association rules and sequential patterns that can be translated to declarative process models later in this thesis.

2 Digitalization of Companies

This chapter is based on a digitalization check by Vossen et al. [VEGD20] that contributes a survey addressing the digital maturity of companies (in particular small and medium-sized enterprises (SMEs)) whose results are then mapped to a multi-dimensional radar chart illustrating the results. Digitalization or digital transformation are very prominent terms, not only in research but also in media and the public.

There are various definitions for digitalization that mean different things or have different foci. One prominent meaning is that it stands for the challenge of adapting procedures and business models of all kinds in a way so that they are compatible with a highly digital, computerized world [25]. This thesis focuses on such definitions of digitalization. In companies and their environment, digitalization is more than the mere transformation of analogue to digital information. Nowadays, there is consensus in research and practice that companies must adapt to the changing environment. If they do not, an organization is likely to get into severe trouble as competitors may develop significant competitive advantages or new challenges may arise that completely change the market field, leaving no room for the established ones anymore [SM17].

A related term is *disruption*. Christensen defines *disruption* as "a process by which a product or service takes root initially in simple applications at the bottom of a market and then relentlessly moves up market, eventually displacing established competitors" [1]. Examples of such a phenomenon are personal computers that replaced mainframes or cellular phones that

displaced fixed-line telephony. Also, the introduction of Apple's iPhone in 2007 is seen as such a disruption because it significantly simplified the process of installing external software (apps) on the phone by introducing the *App Store*. Quite recently, traditional mobility service providers like taxis have been attacked by the likes of Uber and Lyft. These competitors play out their advantages through a digital business model and have significantly better cost and service level structures. Therefore, traditional companies already present in the market should adapt to the new circumstances and change the way they carry out their business model. However, such changes should be planned carefully. Otherwise, a high amount of resources might be spent with no or minimal effect on the digital status of the company.

Therefore, a starting point for digitalization initiatives should be an assessment of the current digital maturity. Researchers and practitioners have developed various approaches. For instance, McKinsey & Company have developed a maturity model that includes several dimensions like strategy, automation, or technology whereby each of them contains a set of subitems that go into more detail [19]. Its purpose is to assess their customer's digital maturity. Academic researchers have, for instance, developed maturity models focusing on particular branches [CBM18, DCMNT17, CWWM14], aspects of digitalization [ASA$^+$19, FAS19], or dimensions [PB18, VL10]. The following presents a digital maturity model that encompasses the most relevant dimension of digitalization and serves as a frame for determining the digital status of a company. It includes the possibility to individually weight each dimension's influence on the total digital value.

2.1 Assessing Digital Maturity

Digitalization is a multi-dimensional concept covering several aspects of a company's operations. Figure 2.1 shows a maturity model for digitalization with seven dimensions. Three dimensions (*Processes*, *Data*, *Business Model*) are primary and four secondary (*Connectivity*, *Interaction*, *Optimization*, *Disruption*) ones. The following introduces each dimension and explains what aspect of digitalization it measures. Then, the characteristics of the survey addressing each dimension through a set of statements are introduced. Afterwards, the survey results must be evaluated by calculating dimension-specific maturity values and a total digital score. Altogether, a radar chart can represent the results best. The relevant dimensions were found by combining related research found by extensive literature research and own considerations.

Processes

Processes are the heart of any company or organization because they directly contribute to the value creation and delivery of the business model. In earlier times, processes were carried out without the use of any Information Technology (IT). With the invention of the computer and the internet, more and more process parts (activities) became digitalized. A fully digitalized process is a process that is entirely executed by the use IT and no or only inevitable manual labour is required. The digital maturity of a company in dimension *Processes* can be rated on a scale between these two extreme cases.

Data

More and more data is available in companies that have to be stored and structured somehow. Similar to processes, in earlier times, data was stored on paper and archived in large physical archives when not required anymore. With the advancing progress in storage capacities, nowadays, the

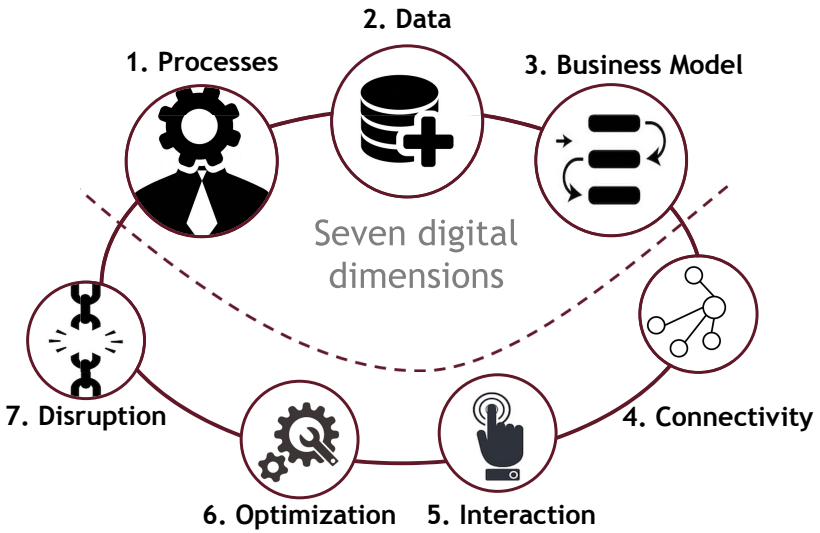


Figure 2.1: Digital Maturity Model with Seven Dimensions [VEGD20].

challenge is not about storing the data itself but rather about generating knowledge about the data and their correct format for use in the processes. The dimension *Data* measures the degree to which the data is digitally available and stored in a structured and harmonized manner.

Business Model

Business models describe how a company plans to earn money from their activities. The goal of this dimension is to measure the degree to which the business model of a company depends on IT and digital technologies. For some business models or branches, a fully digitalized business model is not possible or desirable, e.g., a restaurant sells food in a physical place that cannot be digitalized. Nevertheless, the dimension can provide an

overview of the dependency on such technologies and indicate how the company performs compared to competitors for branches where digital technologies are highly relevant and vital for digital progress.

Connectivity

The *Connectivity* dimension of the maturity model addresses information and communication channels. Thereby, a company achieves a higher score for this dimension with more different channels. Examples of such channels can be mail, chat, or social media channels with customers or internal channels between company departments.

Interaction

While the *Connectivity* dimension solely cares about the quantity of information and communication channels, the *Interaction* dimension evaluates the use of such channels. The digital score for this dimension is based on the intensity of use for digital channels.

Optimization

Optimization refers to the fast-changing property of digitalization and measures the degree to which the company adapts to new digital trends on the market and exploits such opportunities. Examples for such activities are search-engine optimization, process mining, or chatbots.

Disruption

Based on the definition of Christensen [1], the *Disruption* dimension assesses the risk for a company to be disrupted by competitors and to lose its current business model. More digital business models have a lower risk of being disrupted than traditional, analogue business models.

Companies have to conduct a digitalization check to determine the digital status based on these seven digital dimensions. The check in the form of a survey consists of questions and statements addressing the

relevant aspects of each dimension. The following presents some selected types of questions for the *Processes* and *Data* dimension and the calculation model to finally come to an evaluation of a company's current digital maturity.

2.2 A Comprehensive Digitalization Check

To assess the digital maturity for each dimension, specific questions that address the relevant concepts of a particular dimension have to be stated. The responder can either be an employee of the company or an external person, e.g., a consultant. Both options lead to different viewpoints and more or less subjective or objective results. Though, it is noteworthy that the person has to have a certain overview of the company's overall picture. For this reason, the practice has shown that CEOs or at least employees with managerial responsibility are good choices for this survey.

Each dimension of the maturity model has five levels, whereas level 0 indicates the worst and level 4 the best (or most digital) manifestation. The statements are stated in a way that the respondent can reply with options ranging from *Does not apply* to *Applys totally* (5-point Likert scale). Then, an average Likert-scale value of the given answer is calculated representing the digital score for one dimension. Sample statements for the *Processes* dimension look like the following:

Statement:
Many different simple process steps are not digitalized and must be performed manually.

Explanation:
This statement measures whether a company has many activities that are executed without the help of IT.

Statement:
There are processes I do not understand.

Explanation:
Are employees able to capture the overall structure of the processes they work with or do they just perform single activities?

The statements capture various aspects of digital maturity for one dimension. Single statements are not enough, and some overlap in some aspects. However, together they cover a broad spectrum of what is considered relevant for the digital status in one dimension. Also, they can balance outliers caused by weaknesses of the statements or uncertainties in the responses. One statement for the almost equally important *Data* dimension is the following:

Statement:
All information from business processes involving customers are reflected in our database.

Explanation:
Does the company store its knowledge about the customers at a single point?

There are about ten to 15 questions for each dimension. For each of them, a digital score is calculated as the average of the Likert-value given by the respondent. Now that a company's digital status regarding a particular dimension is known, it is clear where the strengths and weaknesses lie. A comparison with branch competitors or similar companies is advisable.

The total digital maturity value that includes all the digital maturity levels of the dimension can deliver a general impression of a company's

status. This value is calculated with a weighted mean of the dimension values. Analysts can adapt the weight of each dimension (a value ranging from 0 to 10) to fit the characteristics of a particular company. The standard setting is a weight of 10 for *Processes*, *Data*, and *Business Model* and 5 for the secondary dimensions. For instance, a craft business may decide that the *Data* or *Business Model* dimension is less critical than dimensions like *Connectivity* or *Interaction* because it has a business model that cannot be fully digitalized but very intense communication with customers and suppliers.

Table 2.1: Sample Calculation of Total Digital Maturity.

Dimension	Score	Weight
Processes	4	10
Data	2.5	10
Business Model	2.5	10
Connectivity	2.25	5
Interaction	1.8	5
Optimization	2.4	5
Disruption	1.7	5
Total Maturity	2.62	-

Table 2.1 shows a sample calculation of the total maturity based on dimension-specific digital maturity values with standard weighting settings. Figure 2.2 visualizes the dimensional maturity values of Table 2.1 using a radar chart. The larger the area of the shape (blue background color), the higher is the overall digital maturity of a company. Assuming a company, for instance, an advertising agency adjudges that for them connectivity and interaction (e.g., for social media communication) is as essential as the three primary dimensions (*Processes, Data, Business*

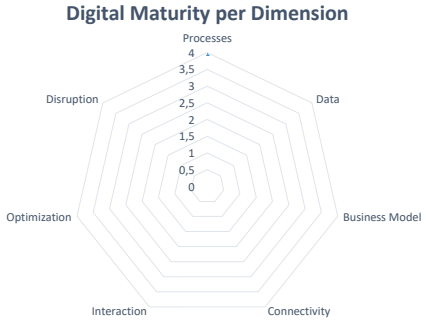


Figure 2.2: Radar Chart for the Maturity Values in Table 2.1.

Model), they can set the dimensions' weight to ten as well. In this case, the total maturity value drops to 2.52. It lies in the self-responsibility of firms to not deceive themselves by setting higher weights for dimensions where they perform well and lower weights for their weak points.

It should be noted that the digital maturity value alone is only a first indication of strengths and weaknesses regarding the digitalization status of a company. Further investigations and comparisons with competitors' results (if known) should be made to judge the result of the digitalization check. Furthermore, the digitalization check does not replace specific consulting and can, as it is, not provide concrete recommendations for actions. The general model for assessing a company's digital maturity is highly relevant for all companies and organizations that want to start their digitalization journey or are already on their way and want to review their progress. This thesis mainly focuses on the *Processes* dimension, which is one of the primary digital dimensions (cf. Figure 2.1), most likely the most important one.

2.3 Processes as a Digital Dimension

Processes are the heart of any business activity. They describe how companies earn profit and how their workflows look like. The business processes influence nearly every other aspect of the company structure or performance. One possibility is to understand the digitalization of processes as transforming a formerly analogue activity to a digital one. However, there are different kinds of "digital" processes and activities:

For instance, one of the statements of the digitalization check shown previously addresses the automatization aspect. In the lowest maturity level, a process step is entirely manual. A person executes a process step without any support from IT systems. Another type of digital process activity is an activity that is performed by humans but supported by IT systems. An example may be a formerly paper-based form that is now digitally available and is filled out by an employee with a PDF editing program. More advanced forms of digital process steps are executed fully or partly automatically, such as script or batch jobs. Such semi- or fully automated process steps require only little human intervention if something goes wrong or a state appears that cannot be solved by a program. An example for such a process step is the import of an invoice in PDF format by an Optical Character Recognition (OCR) application that only requires human intervention when, for instance, an invoice field contains data in the wrong format. Often, they represent the ultimate goal for a business process digitalization initiative when no full automation without any human interaction is possible or desired.

Fully-automated process steps such as the execution of database or financial transactions do not require any manual work. In case of an error, they can automatically recover a functional state. Process steps can be ranged in the described spectrum with zero representing fully manual and four fully automatic activities whereby a continuous scale is used.

For optimal results, however, processes should not only be fully digitalized and automated but also optimized regarding structure and control flow. In this context, research and practice often use the notion of "streamlined" processes. There is a whole field in research and business that focuses, inter alia, on the design and implementation of business processes: Business Process Management (BPM). The following gives an overview of BPM and motivates the use of process mining as one of the most promising technologies in the process analysis research discipline for current and future process analysis. Progress regarding a company's business processes directly contributes to improving the digital maturity status in the *Processes* dimension. Therefore, Chapter 2 serves as broader motivation for the main aspects and contributions of this thesis.

3 Business Process Management and Process Mining

Dumas et al. define BPM as "the art and science of overseeing how work is performed in an organization to ensure consistent outcomes and to take advantage of improvement opportunities" [DLRM+13]. It is noteworthy that the discipline does not focus so much on the transformation or change of individual activities but rather on a holistic view and management of organizational processes that add value for the company and customers. A major part of BPM deals with capturing and modeling business processes. For this purpose, several notation forms have been introduced in the past decades that can capture the shape and characteristics of processes. Often, the goal is to create an evaluation of the process situation, identify weaknesses and potentials for improvement and finally adapt the real-world process.

Vossen and Lechtenbörger [VL19] collect different approaches for process modeling, namely the Event-driven Process Chain (EPC), the Business Process Model and Notation (BPMN) and the Petri net notation. This thesis focuses on Petri nets as one well-known and proven representative of process modeling languages. Schönthaler et al. [SVOK12] propose the Horus method as an approach for covering all aspects related to business processes and their integration in and combination with the residual business parts. Its procedure modeling is based on an adapted version of Petri nets. The following presents the concept of the Horus method.

3.1 The Horus Method

The Horus method serves as a frame around business (process) analysis and leads from project preparation to application. Figure 3.1 shows an overview of the Horus method with its phases, steps, and accompanying activities. Phase 0 prepares the projects, sets goals, and produces a project initialization and definition. The *Strategy and Architecture* phase (Phase 1) includes an assessment of strategic and system architecture aspects. Phase 2 (*Business Process Analysis)* represents the core of the Horus method in which several models addressing organization structure and risk analysis are created. Business processes and the development of the procedure models using the adapted Petri net notation stand in the middle of all analyses here. Finally, Phase 3 (*Application*) encompasses the actual use and implementation of the previously developed models and analyses.

The Horus method represents a well-proven reference model for BPM projects in the industry. Usually, consultants go into a company and analyze the as-is state in respect of the aspects shown in Figure 3.1. Thereby, she or he conducts interviews with employees (e.g., process owners) who can deliver relevant information, especially about the procedures, and then creates the models. The Horus Business Modeler [10] supports creating all models included in the Horus method.

Naturally, interviews result in partly subjective impressions. Therefore, an overall goal may be to automatize capturing the analyses of the Horus method by applying specific algorithms on objective data. At the present time of this thesis, there are no approaches for the automatic generation of most aspects of the Horus method. However, process mining has arisen to discover procedure models based on event data input. This technology could be integrated into the *Business Process Analysis* phase of the Horus method, complementing the process modeling activities. Consequently,

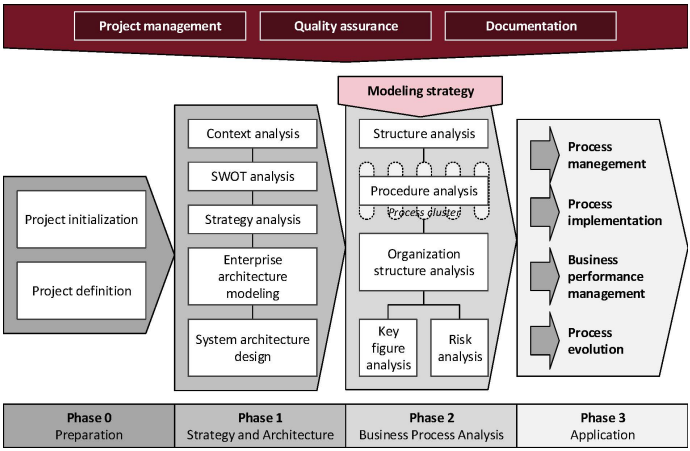


Figure 3.1: Overview of the Horus Method (Source: adopted from [SVOK12]).

Business performance management could be automated as process mining can constantly monitor the actual executions. Recommended adaptations of the process (*Process implementation and evolution*) can also be derived from process mining relatively easily.

3.2 Imperative Process Modeling

Imperative or procedural process modeling denotes a paradigm where a process is modeled by a graph describing a flow of activities. It aims to create an abstraction of the real-world business using a graph that shows the allowed control flow. In general, imperative approaches focus on the "How"; they define how concrete instances of something should look like. Note that here imperative does not fully comply with the meaning in the

context of programming languages. Fahland et al. [FLM+09] elaborate on this analogy in more detail.

Examples for such notations are EPCs, BPMN, or Petri nets. BPMN is standardized and maintained by the Object Management Group (OMG) [2]. Figure 3.2 shows a small payment process in the BPMN notation. In the most simple form, BPMN includes symbols for tasks, gateways, sequences, and events. The sample process starts with a start event that leads to an activity *Identify payment method*. Then, a decision regarding the payment method is made. If the customer wants to pay by check or cash, the next activity is *Accept cash or check*. *Process credit card* follows in case credit card was selected as the payment method. The *Payment method?* gateway represents a logical XOR-decision. Both alternative flows eventually lead to the *Prepare package for customer* activity. It is noteworthy that no AND-gateway precedes this activity. Thus, one of the input flows is enough to continue the process. This matches the XOR-decision before the two processing activities. Finally, the process is terminated by an end activity depicted as a circle with a bold border.

BPMN is a de facto standard for process modeling in practice and has a large set of symbols allowing users to develop comprehensive and complex process representation. Despite its powerfulness and wide adoption, the BPMN notation is criticized for having too many symbols leading to an impaired understanding of the models.

Petri nets are an alternative notation that uses a minimum set of symbols but can express any process in a comprehensible way, originating from the Ph.D. thesis of Petri in 1962 [Pet62]. Figure 3.3 shows an equivalent representation of the process model in Figure 3.2 in an (adapted) Petri net notation in the Horus Business Modeler [10]. The places represent data objects that are provided as an input for the activities or are their output. Activities (represented as rectangles) modify the input objects or create new ones. Activity elements indicate XOR-decision paths (both

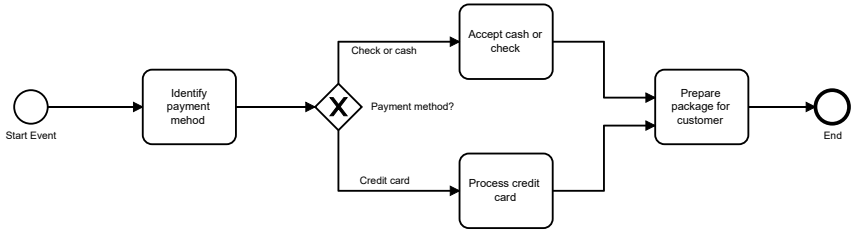


Figure 3.2: Sample Process Model in BPMN Notation (Source: adopted from [3]).

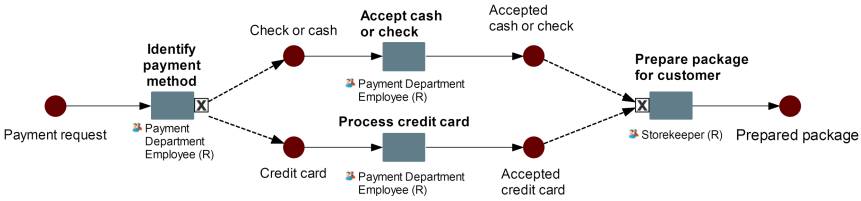


Figure 3.3: Payment Process in the Horus Business Modeler [10] (based on Figure 3.2).

input and output) with an X before or after the rectangle. Here, Horus uses a slightly adapted Petri net notation to enable the active decision for exclusive choices. The same model without the extended activity rectangles would depict a parallel control flow where both inputs are required for the *Prepare package* activity.

The Horus method (like BPMN), is based on the imperative process modeling paradigm. It uses an adapted version of the Petri net notation to create procedure models of business processes. However, researchers have investigated the declarative process modeling paradigm. With that approach, processes are no longer represented by a control flow graph but

by constraints and conditions that restrict the executions of the process. Everything that is not specified by the constraints is allowed, comparable to an "open world assumption" in the database context. This modeling style is helpful in cases where the imperative modeling of highly-flexible processes would result in spaghetti-like diagrams or only conditions, but no concrete information about the procedure of the process is known. The following introduces the declarative process modeling paradigm in more detail and illustrates the historical early days until today's use in the process mining discipline.

3.3 Declarative Process Modeling

Declarative approaches, in general, have the property that they do not prescribe how something should be constructed or implemented on a detailed level. Instead, they describe properties and conditions that have to hold. The concrete shape of the realization is left open. Examples in the programming language context are SQL or Haskell. For process modeling, this means creating a model in the form of a graph describing the (allowed) flow of the process. Imperative notations completely describe the allowed control flow of the process. Everything else that is that not specified by the model is not part of the process. Declarative notations, on the contrary, forbid certain behavior that is not in line with the constraints. Additionally, some declarative notations include constraint templates that allow the definition of optional behavior, for instance, the (exclusive) choice between two activities. Everything else that is that not captured by the constraints set is allowed.

Imperative approaches (e.g., BPMN or Petri nets) have the advantage that they are relatively easily understandable. Users of the process model can directly see which process paths are desired and which are not and therefore identify deviations. At the same time, experience has shown that

declarative approaches have advantages for large, complex, and loosely structured processes with many variants. Modeling such processes with imperative approaches can result in huge process models, hardly understandable diagrams with many paths, especially when not using abstraction layers with super- and subprocesses. Investigations about the understandability of both paradigms have, for instance, been made by Fahland et al. [FLM+09] or Pichler et al. [PWZ+11].

Thus, alternatives and solutions for this problem have to be found. Declarative process modeling comes into play here. This paradigm alone is not new, as contributions in research were already made 20 years ago. For instance, Desel and Erwin [DE00] have addressed the topic of hybrid (combining imperative and declarative model elements) process specifications for workflow systems. Van der Aalst et al. [vDAPS09] contributed the Declare framework in 2009. More recently, in 2016, Slaats et al. defined formal semantics for hybrid process modeling notations [SSMR16]. With process mining emerging, declarative approaches have got a new application field.

Like imperative approaches, declarative ones also need notations used to express the process model. One of the most widespread notations for declarative process models in research is Declare. Pesic et al. present Declare as a constraint-based, declarative approach that uses instances of constraint templates to model a process [PSVdA07].

Constraint templates in Declare are based on Linear Temporal Logic (LTL) [GPC99]. It is a temporal extension of first-order logic including temporal operator such as (*eventually* ($\Diamond$), *always* ($\Box$), *until* ($\sqcup$) and *next time* ($\circ$)). The expressive power is equivalent to first-order logic with monadic relation symbols and the smaller relation (<). Systematically relating the logics to each other, a superset of LTL and the computation tree logic (CTL) forms the CTL* logic.

Figure 3.4 shows some Declare constraints with an explanation and graphical depiction [ADCH+]. Unary constraints are applied to single activities, whereas binary constraints define a relationship between two activities. For instance, the **EXISTENCE** constraint prescribes that a process instance has to include an activity at least once. **INIT** or **END** constraints enforce that some activity is first or last for every execution of a process.

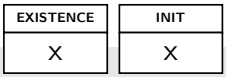
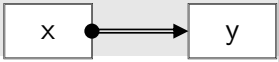
Template	Explanation	Notation
Unary constraints		
EXISTENCE (x)	Activity x occurs at least once per trace.	
INIT (x)	Activity x occurs at the beginning of every trace.	
Binary constraints		
RESPONDED EXISTENCE (x,y)	If x occurs, then y must occur before or after x.	
RESPONSE (x, y)	If x occurs, then y must occur eventually after x.	
CHAIN RESPONSE (x, y)	If x occurs, then y must occur immediately after x.	
PRECEDENCE (x, y)	y occurs only if preceded by x.	

Figure 3.4: Sample Declare Constraints (based on [ADCH+]).

A binary **RESPONDED EXISTENCE** constraint between two activities indicates that whenever an activity x is executed, activity y also has to be part of the trace, i.e., it has to occur before or after x. The **RESPONSE** constraint, though, takes the temporal order into account and demands that if one activity appears, another activity has to be executed at some point in time subsequently. It can also be that the first activity is not part of the trace. In this case, the constraint is trivially (or vacuously) satisfied.

Using an LTL formula, the **RESPONSE** constraint is described as follows: $\Box(A \rightarrow \Diamond B)$. **CHAIN RESPONSE** constraints are strengthened versions of this type and prescribe that the second activity has to be executed directly afterwards. **PRECEDENCE** constraints are in some way the counterpart of **RESPONSE** in that they enforce that an activity is only executed if some other activity was part of the trace beforehand.

There are more constraint types, like, for instance, **SUCCESSION**, that enforces a direct succession of two activities or **EXACTLY_N** with N a number greater or equal to one that prescribes that an activity appears exactly N times in a trace. Together, the constraints form a description of a process that matches representations in imperative modeling notations like BPMN or Petri nets. Synchronization or parallelism in the style of procedural models is not directly supported by the standard Declare set; however, the binary **CHOICE** constraint can specify that either one or both activities together have to occur. **EXCLUSIVE CHOICE** constraints demand that either one or another activity occur but not both together. Declare constraints forbid behavior that is not in line with the constraints. Only some of them, for instance, the **CHOICE** or **EXCLUSIVE CHOICE** templates, leave some optional freedom to the process. An extensive overview of the set of Declare templates can be found in the appendix in Chapter A.

Generally, it is theoretically possible to describe any process using imperative, declarative, or hybrid specifications. Figure 3.5 presents a declarative representation of the payment process in Figures 3.2 and 3.3. Note that the Declare model does not explicitly contain all coherences of the imperative representation, often there is no 1-to-1 translation, and various variants are possible. For instance, it does not prescribe that *Prepare package for customer* has to directly follow *Accept cash or check* or *Process credit card*. Theoretically, other activities could occur in between. Also, the model does not indicate that either *Accept cash or check* or *Process*

credit card occur, and no parallel execution of them is allowed. In this case, these losses of information can be prevented. First, introducing a **NOT CO-EXISTENCE** constraint between the *Accept cash or check* and *Process credit card* activities and second, changing the **RESPONSE** to **CHAIN RESPONSE** constraints aligns the declarative and imperative model. However, especially for such smaller process representations, modelers commonly face a trade-off between the complexity and size of the model, the loss of information, and the additional freedom that shall be granted to the process.

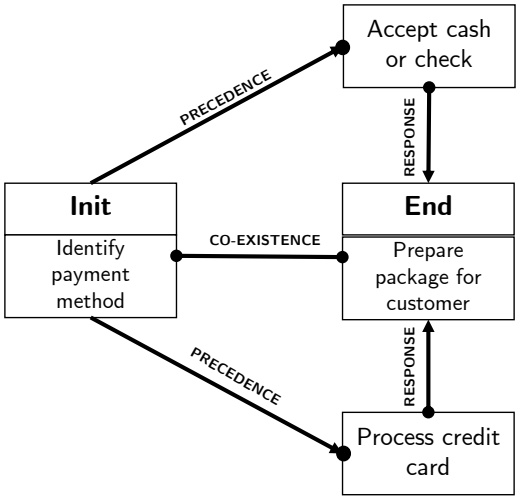


Figure 3.5: Declare Representation of Payment Process (Figures 3.2 and 3.3).

The Declare notation does not have the same expressive power as imperative modeling notations. One reason is that even though it allows the definition of a direct control flow between two activities through

the **CHAIN RESPONSE** constraint type, parallelism and exclusive or inclusive joins of activities cannot be represented. The **EXCLUSIVE** and **CHOICE** constraints in the Declare template set are not suitable for at least three activities which would be required to represent a split or join operation. Still, the declarative specification of process models can be extended with data conditions. Thereby, the model additionally contains conditions and constraints on the attributes related to the activities. One representative of this concept is Multi-Perspective Declare (MP-Declare). The following introduces this extension of Declare and points out the relevant types of conditions for this thesis.

3.4 Multi-perspective Declarative Constraints

Multi-perspective declarative process modeling denotes the enrichment of declarative process models with data-aware conditions. Westergaard et al. [WM12] and Masellis et al. [DMMM14] contribute groundwork by examining the static check of temporal constraints regarding their consistency and the monitoring of data-aware constraints on process event streams. This thesis mainly takes Declare as the declarative process model representation of choice, therefore it seems natural to focus on one of its most prominent multi-perspective extension MP-Declare.

Burattin et al. [BMS16] formalize MP-Declare. They use Metric First Order Temporal Logic (MFOTL) as the basis to define the semantics of the conditions that it can construct. The authors present polynomial-time algorithms to check the conformance of multi-perspective declarative models and event data. This is in line with the finding that MFOTL is decidable for finite sets of words [BKMZ15] which is the case for event logs. MFOTL is an extension of metric temporal logic (MTL) for which the satisfiability problem is EXPSPACE-complete meaning that it can be solved by a deterministic Turing machine in exponential space [Koy90].

Multi-perspective extensions of Declare, in general, intend to use additional data perspectives for the definition of constraints in declarative process model descriptions. The assumption is that one or more attributes of different types (e.g., Strings, Integers, Boolean, etc.) belong to an activity. Taking these attributes, MP-Declare can express three types of conditions, namely activation, correlation, and time conditions.

The first activity of a binary constraint is called *activation* and the second one *target*. Activation conditions can be applied to both unary and binary constraints. They describe a condition that has to be fulfilled to make a condition active, i.e., if it does not hold, the constraint cannot violate the Declare model no matter how the actual process execution looks. This thesis uses a slightly different notion of activation conditions compared to Maggi et al. [MMB19] in the sense that they can also include attributes of the target activity. Hence, the activation condition applies quasi for the full constraint and not only the activating activity. An attribute named *resource*, for instance, is accessed as *A.resource* or *T.resource* depending on whether it is part of the activation or target activity. Correlation conditions define relationships between attributes of two activities. They can only be applied to binary constraint types and have to hold when the target occurs [MMB19].

Time conditions introduce additional, more concrete temporal constraints between two activities. It is possible to specify a time window in which an activity has to be executed after another. For instance, a time condition *[2,5,h]* applied to a **RESPONSE** constraint between two activities means that the second activity has to be executed between two and five hours after the first one.

Figure 3.6 enriches the declarative model in Figure 3.5 with activation, correlation, and time conditions. For unary constraints, activation conditions are represented inside brackets after the constraint's name. Binary constraints have three brackets where the first one indicates the

activation, the second one the correlation, and the third one the time condition.

The **PRECEDENCE** template between the *Identify payment method* and *Accept cash or check* activities has no activation condition. The same person must execute both activities (correlation condition *[same resource]*). Accepting the cash or check should not take place later than two days after the identification of the payment method (time condition *[0,2,d]*. The activation condition of the **INIT** constraint on the *Identify payment method* activity indicates that only in case the price is below 100, the payment method has to be identified first. It could be, for instance, that all payments with a price above must be paid with a credit card. Therefore, the identification of the payment method is obsolete. To make sure that in this case, the processing of the credit card can take place independently, the other **PRECEDENCE** constraint requires an activation condition *A.price < 100*. Time conditions are not further considered as the approach of this thesis cannot discover them from event logs.

Besides manually constructing declarative models, they could also be derived automatically by applying process mining to event logs. The following introduces process mining and puts a special focus on the discovery type as the main focus of this thesis.

3.5 Process Mining

The foundations for process mining itself as known today were laid in 2011 with the Process Mining Manifesto [vdAAdM+12] although the basic concepts were already known and discussed before, e.g., by van der Aalst [VdAVDH+03, VdAWM04]. It is based on the idea that process executions are supported by information systems that leave traces, e.g., database entries, message interchanges, or log files. Overall, the goal of process mining is to analyze business processes based on their real-

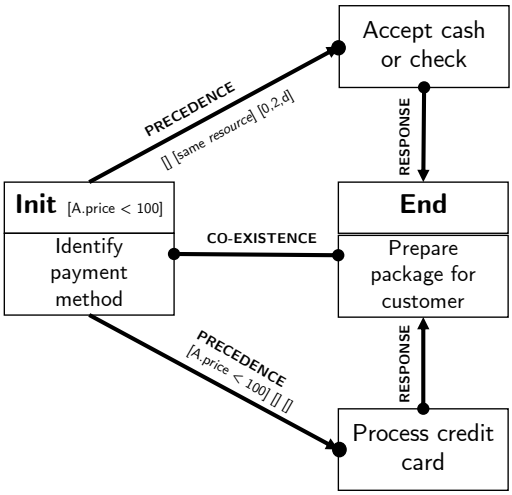


Figure 3.6: Multi-perspective Declare Model of Payment Process (based on Figure 3.5).

world executions. Process mining stands in contrast to traditional process modeling activities where a process is constructed based on observations and interviews with process stakeholders.

The following introduces the history and foundations of process mining and shows the relationships to related research fields, including a delimitation of the discipline. Then, a glimpse into the application of process mining in practice, including reference processes for a best-practice-oriented way of proceeding, follows. Event data in a suitable format (event logs) is the crucial input component for any process mining initiative. Hence, the requirements for such event logs and possible ways to construct them are presented. Afterwards, the discovery type of process mining is introduced with the standard algorithms and KPIs description. Conformance checking

represents another essential target of process mining applications. Finally, the thesis sheds light on process mining with the declarative modeling paradigm and shows differences to classic imperative applications.

3.5.1 History and Foundations

Although the concept of process mining itself is only around ten years old, there had been research about similar concepts and terms years before. For instance, van der Aalst et al. addressed the issue of "workflow mining" that has very similar properties and can be seen as a predecessor of process mining in 2003 [VdAVDH+03]. In 2004, van der Aalst et al. presented one of the first algorithms to discover a process model based on given execution data [VdAWM04]. Generally, process mining is positioned at the intersection between BPM and Data Science [Wil16]. Data Mining or analysis methods are now also applied to process event data. At the same time, it remains a process analysis task to improve and optimize existing business processes. Figure 3.7 visualizes these relationships. Van der Aalst et al. [vdAAdM+12] classify process mining as a subset of process intelligence which in turn is a subset of Business Intelligence (BI). Questions regarding the integration of process mining into existing company structures arise. For instance, when applying process mining in practice today, it has to be decided whether the competencies and skills should be pooled centrally, e.g., at a BI department or a so-called Center of Excellence (CoE) or distributed in a decentralized manner.

Another frequent challenge of process mining is transforming and integrating the relevant data into a structured file, a so-called event log. In contrast to process modeling, process mining allows the detection of AS-IS processes without or very little bias compared to reality. The bias in the process modeling cases can originate from the interviews with employees of an organization who are part of the business process under analysis.

For instance, it can be the case that the employee does not remember all details correctly or has a subjective view on things that could cast a poor light on her or him.

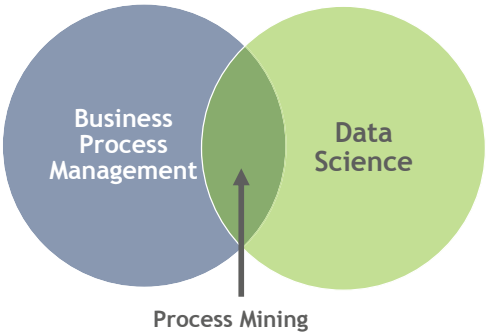


Figure 3.7: Relationships of BPM, Data Science and Process Mining (based on [Wil16]).

Primarily, process mining consists of three types [Wil16]: Process discovery denotes the type that focuses on the generation of process models from event data input. No, or nearly no knowledge about the actual execution of the process is required. However, it is necessary to know how to map (raw) data excerpts to activities of the process under consideration. With these mappings, it is possible to create an event log suitable for process mining tools. The conformance checking type covers all aspects of checking whether expectations about the process match with reality. Thereby, among other things, pre-defined, hand-made process models are compared to either process models created by discovery algorithms or the executions themselves. Process enhancement varies this approach. Process models are adapted based on what process mining finds during the analysis of the process event data. Overall, the goal is to have a pro-

cess model that describes and matches the actual execution as closely as possible. The following elaborates on the fundamental terms and concepts of process mining, starting with event data as the input for all process mining activities.

3.5.2 Event Data

Event data denotes data that describes the execution of a process in a business or any other organizational environment. It is a crucial input for any process mining activity. Because business processes today are supported by various information systems that are possibly part of many departments and organizations, all relevant information must be collected and integrated into an event log beforehand. In the best case, a process at hand is modeled and implemented in a workflow management system that allows the generation and export of logs. However, many business processes have developed historically without using a central instance like a workflow management system.

Instead, the relevant event data is distributed over many systems and cannot be collected from a single source [Wil16]. Moreover, the information about the execution of a particular activity of a process may not be represented explicitly but may come in the form of database entries or change logs, log files, or Electronic Data Interchange (EDI) messages. In these cases, the challenge is to transform and map the data pieces to concrete activities of process instances. Thereby, humans constructing the event log require a certain amount of domain knowledge about the process. Until today, the preparation of the process event data remains a significant challenge and hurdle for process mining [Wil16].

Dakic et al. [DSL+19] perform a survey and literature review that result in 15 publications contributing approaches for the extraction of event data from information systems. They differentiate between Process-

Aware (PAIS) and Process-Unaware Information Systems whereby Dumas et al. [DVdATH05] define PAISs as systems that are implemented with the knowledge that they support the execution of a series of activities described by a process model. Non-PAISs represent the opposite and are often historically developed legacy systems introduced into an existing IT landscape without considering the overall process. The extraction processes differ depending on the source systems; research has contributed several approaches.

In practice, frequent sources of process event data are data and object-specific ERP systems that often hold the relevant event data in database tables. De Murillas et al. [dMRVDA19] propose a generic approach for extracting event data from databases. For that purpose, they introduce a metamodel compatible with the XES format that provides structured storage capabilities for all information related to processes and their activities. Extracting event data is performed by using the SQL language. The authors provide three sample types of sources in database format: Redo logs of database tables, in-table versioned tables, and SAP style change tables. Table 3.1 shows an excerpt of an in-table versioned table.

Lines 1 and 2 and lines 3 and 4 show entries for address data with different timestamps. For the first entry, the *name* and for the second, the *address* attributes are different, although they refer to the same id. This in-table versioning of changes indicates that some process activity has changed these values. Changing values can be mapped to an execution of a particular activity. Using domain knowledge, the changes could, for instance, be traced back to *Update name* and *Update address* activities of a process that updates the master data of a customer.

Another standard storage format for in-table versioned process data is SAP-style change tables. Because SAP is widely used as an ERP system, more and more commercial process mining vendors develop adapters and connectors for these systems. The overall goal is to establish standard

routines for extracting event data from SAP database tables and lower the barriers for introducing process mining [dMRVDA19].

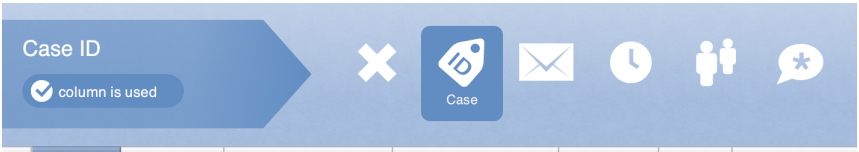
Castillo et al. [PCWP+11] contribute an approach for extracting event data from non-process-aware information systems. With the proposed technique, code inspection of Legacy Information Systems (LIS) can obtain an event log just like for PAISs. All in all, the extraction of event data remains a significant challenge for process mining that still requires substantial work and efforts [Wil16].

Table 3.1: In-table Versioning of Personal Data (Source: adopted from [dMRVDA19]).

id	load_timestamp	name	address	birth_date
17299	2014-11-27 15:57:08.0	Name1	Address1	01-AUG-06
17299	2014-11-27 16:07:02.0	Name2	Address1	01-AUG-06
17300	2014-11-27 17:48:09.0	Name3	Address2	14-JUN-04
17300	2014-11-27 19:06:12.0	Name3	Address3	14-JUN-04

Nearly all process mining tools available today require an event log with three mandatory attributes as input for every line: A *case ID* uniquely identifying a trace of the process, an *activity name* denoting the execution of a particular activity, and a *timestamp* that provides the information about the activities' execution order. With these three attributes, process mining tools can discover a process model. It is possible to add more optional attributes like resources or costs for a more in-depth analysis of Key Performance Indicators (KPIs). Figure 3.8 shows a screenshot of the import process for a sample event log in Disco [Wil16, 4]. A role is assigned to each column. The event log is in CSV format and describes an insurance compensation request process. Disco is a standalone process mining tool originating from academia. The event log includes three

mandatory (*Case ID*, *Activity*, and *dd-MM-yyyy:HH-mm*) and two optional attributes (*Resource* and *Costs*).



	Case ID	Event ID	dd-MM-yyyy:HH.mm	Activity	Resource	Costs
1	1	35654423	30-12-2010:11.02	register request	Pete	50
2	1	35654424	31-12-2010:10.06	examine thoroughly	Sue	400
3	1	35654425	05-01-2011:15.12	check ticket	Mike	100
4	1	35654426	06-01-2011:11.18	decide	Sara	200
5	1	35654427	07-01-2011:14.24	reject request	Pete	200
6	2	35654483	30-12-2010:11.32	register request	Mike	50
7	2	35654485	30-12-2010:12.12	check ticket	Mike	100
8	2	35654487	30-12-2010:14.16	examine casually	Sean	400
9	2	35654488	05-01-2011:11.22	decide	Sara	200
10	2	35654489	08-01-2011:12.05	pay compensation	Ellen	200
11	3	35654521	30-12-2010:14.32	register request	Pete	50
12	3	35654522	30-12-2010:15.06	examine casually	Mike	400
13	3	35654524	30-12-2010:16.34	check ticket	Ellen	100
14	3	35654525	06-01-2011:09.18	decide	Sara	200
15	3	35654526	06-01-2011:12.18	reinitiate request	Sara	200
16	3	35654527	06-01-2011:13.06	examine thoroughly	Sean	400
17	3	35654530	08-01-2011:11.43	check ticket	Pete	100
18	3	35654531	09-01-2011:09.55	decide	Sara	200
19	3	35654533	15-01-2011:10.45	pay compensation	Ellen	200
20	4	35654641	06-01-2011:15.02	register request	Pete	50
21	4	35654643	07-01-2011:12.06	check ticket	Mike	100
22	4	35654644	08-01-2011:14.43	examine thoroughly	Sean	400
23	4	35654645	09-01-2011:12.02	decide	Sara	200
24	4	35654647	12-01-2011:15.44	reject request	Ellen	200
25	5	35654711	06-01-2011:09.02	register request	Ellen	50
26	5	35654712	07-01-2011:10.16	examine casually	Mike	400

Figure 3.8: Import of Insurance Compensation Request Process Event Log into Disco [4].

An alternative storage format for event logs was introduced by Verbeek et al. [VBVDVDA10] in 2010. The *eXtensible Event Stream (XES)* format addresses the weaknesses of the previously used *MXML* format. It resulted in the "IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams" in 2016 [Gro16]. Like

MXML, XES is XML-based. Listing 3.1 presents an equivalent representation for one event of the event log excerpt in Figure 3.8 in *XES* format. Figure 3.10 shows the underlying process model resulting from discovery in Disco.

First, the log defines the required extensions. The *Lifecycle*, *Time*, and *Concept* extensions are necessary to specify the activity name, the timestamp and the status of the corresponding activity (e.g., scheduled, started, assigned, completed, etc.). Then, attributes on both the event and trace level are defined whose values are taken as standard whenever they are not defined by the traces and events themselves [VBVDVDA10]. Line 16 gives a name to the event log. Inside the *<trace>* tags, a name is given to a specific trace (line 18), and line 19 assigns a value to the case attribute *totalValue* indicating the sum of all activity costs in a particular trace.

Finally, lines 21 to 25 determine the name, status, resource, timestamp, and costs of the *Register request* activity. The *XES* file uses all extensions introduced in lines 2 to 6 and assigns concrete values to attributes of the extensions. Even though research-inspired process mining tools like ProM [7] or Disco [4] and commercial ones like Celonis [5] support the *XES* event log format, the CSV format is probably more common in practice because it is a well-known, accepted standard for information interchange as against XES being a standard originating from research activities.

Listing 3.1: Excerpt of Insurance Compensation Request Process Event Log in XES Format.

```
1 <log>
2 <extension name="Lifecycle" prefix="lifecycle" uri="
      http://www.xes-standard.org/lifecycle.xesext"/>
3 <extension name="Time" prefix="time" uri="http://www.
      xes-standard.org/time.xesext"/>
4 <extension name="Concept" prefix="concept" uri="http://
      www.xes-standard.org/concept.xesext"/>
```

```
<extension name="Organizational" prefix="org" uri="
    http://www.xes.standard.org/org.xesext"/>
<extension name="Costs" prefix="cost" uri="http://www.
    xes-standard.org/cost.xesext"/>
<global scope="trace">
<string key="concept:name" value="unknown"/>
</global>
<global scope="event">
<string key="concept:name" value="unknown"/>
<string key="lifecycle:transition" value="unknown"/>
<string key="org:resource" value="unknown"/>
</global>
<classifier name="Activity classifier" keys="
    concept:name lifecycle:transition"/>
<string key="concept:name" value="Compensation request
    log" />
<trace>
<string key="concept:name" value="Compensation Request
    1" />
<float key="costs:totalValue" value="950" />
<event>
<string key="concept:name" value="Register request" />
<string key="lifecycle:transition" value="complete" />
<string key="org:resource" value="Pete" />
<date key="time:timestamp" value="30-12-2010:11.02"/>
<float key="cost:currentValue" value="50" />
</event>
</trace>
</log>
```

Case IDs can have any format (not only numerical) as long as they are the same for any activity that belongs to a process instance. The screenshot in Figure 3.8 shows four different *Case IDs* meaning that it describes four executions of the same process. The full event log contains six. Each *Case ID* appears in a line of the log from four to nine times,

depending on the number of activities that were part of one particular process instance. All activities should have descriptive names and must be equal for every entry that relates to the same activity (e.g., *register request* in every case and not *register_request* in some of them).

The time information must be in timestamp format. Theoretically, only the information about the date has to be given. However, it is advisable to specify it with at least minute-by-minute precision in practice. The timestamp should provide second or even millisecond information for processes with shorter intervals between the activities.

Additionally, optional attributes can be divided into two types: *Case attributes* are related to a case and are equal for each activity of the same process instance. *Activity* or *event attributes* are related to one activity of the case and can have different values for each activity of the same process instance. The *Resource* and *Costs* attributes in Figure 3.8 are activity attributes and specify the employee (human resource) who was assigned to a particular step of the insurance compensation request process and how high the costs for each step were. An example of a case attribute could be the total amount of an order handled by an *Order2Cash* process. One distinctive feature or indication for case attributes (in CSV files) is that the value is equal for all activities related to the same process. In general, activity attributes are more frequent than case attributes because they go well with the typical structure in CSV files.

Process discovery is one of the three types of process mining (most likely the most prominent) and deals with generating process models from event log input. The following provides more details, including a concise comparison of discovery algorithms in the literature.

3.5.3 Process Discovery

Process mining discovery takes an event log (for instance, in CSV or XES format) as an input and generates a process model underlying the execution data. Several mining algorithms have been contributed to the literature to perform such transformations. Van der Aalst [Wil16] identifies four competing quality criteria of process discovery algorithms: *fitness*, *precision*, *generalization*, and *simplicity*.

Fitness denotes the ability of a process model to represent as many traces of the event log as possible. The *precision* criterion indicates the property of a resulting process model to specifically match the behavior in the event log. The *generalization* indicator addresses the opposite issue indicating whether it overfits the event log. Finally, the *simplicity* criterion favours simpler, more compact process models over complex, confusing ones. These four criteria are affected by each other and can together evaluate the result of the process discovery process [Wil16]. There are several algorithms, each of which focuses on one particular criterion (e.g., generalizability) while neglecting others.

The most common and straightforward process discovery algorithm is the α-algorithm, first introduced by van der Aalst et al. [VdAWM04] in 2004. Its fundamental idea is to create ordering relations (directly follows, causality, not directly follows, parallelism) between activities transformed into a "footprint matrix". Table 3.2 shows such a matrix whereby *a*, *b*, and *c* denote the activities in the event log. The # symbol indicates no direct sequence, $\rightarrow$ and $\leftarrow$ direct sequence in one of the directions and || a concurrency relation between the activities. The α-algorithm then takes these footprints and creates sequences, XOR-splits, XOR-joins, AND-splits, and AND-join patterns. For instance, the ordering relations $\rightarrow$ b, a $\rightarrow$ c and b || c together in Table 3.2 can be transformed to an AND-split pattern like depicted in Figure 3.9. The entirety of such transformations and patterns forms the final process model.

Table 3.2: Sample Event Log Footprint Matrix (Source: adopted from [Wil16]).

	a	**b**	**c**
a	#	#	→
b	←	#	||
c	←	||	#

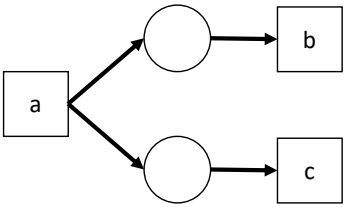


Figure 3.9: AND-split in Petri Net Notation (Source: adopted from [Wil16]).

The α-algorithm works well with larger event logs and can deal with parallel events. Furthermore, it nearly always produces a process model with a high fitness to the event log. However, it has difficulties dealing with noise, loops, and infrequent behavior in the log [Wil16]. Therefore, several improvements have been developed to overcome these weaknesses. Besides, researchers have developed heuristic, genetic, region-based, inductive, and other process discovery procedures and algorithms.

Weijters and Ribeiro [WR11] contribute a heuristic process discovery approach. Heuristic approaches use causal nets and calculate a dependency value between the activities based on the frequency of the "directly follows" relation. With that values, a dependency graph is constructed that is then used to generate splits and joins for the resulting process model while considering certain threshold values related to the frequency of

the relation. Because the heuristic algorithms take the frequencies into account, these algorithms are resilient against infrequent behavior in the event log. In sum, heuristic approaches are well suited for users who want to achieve high precision of the event log and process model.

Leemans et al. [LFvdA13] propose an algorithm for inductive process mining. Inductive mining techniques split event logs into sub logs, organize them into process trees, and finally derive process models. Today, there are various robust and proven algorithms for process discovery. If possible, users who apply process mining should try out various ones to find the algorithm that suits their needs (performance criteria) and event data best.

Popular process mining tools like Celonis [5], or Disco [4] implement some of these process discovery algorithms. Often, however, they do not offer the explicit possibility to choose between different mining algorithms. Most of the tools can vary the percentages of paths and activities of the resulting process model shown based on the frequency with which they appear in the traces. Figure 3.10 shows the output for a process discovery in Disco [4]. The process model shows a compensation request process in an insurance company based on the event log in Figure 3.8. Absolute frequencies of the activity occurrences are provided in the activity boxes, whereas the absolute frequencies of the paths are placed at the respective edges. Only the most common paths are shown, but the relative frequency of shown activities is set to 100%.

Figure 3.11 depicts the same process discovery but with the percentage of paths set to 100%, too. The process model now represents all traces of the event log. However, even for this small sample, the diagrams become more complex and harder to understand. Users of process mining tools like Disco can play around with the settings and concentrate on the parameters and indicators that fit their desired analysis best. Other metrics are, for instance, the case frequency (number of cases an activity appears in),

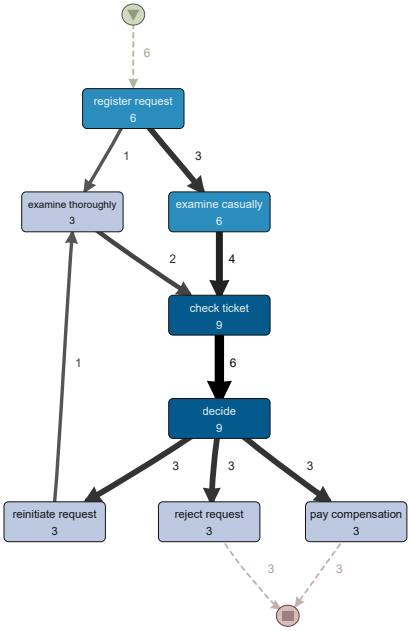


Figure 3.10: Discovery of Compensation Request Process in Disco [4].

the total duration, or the lead times of the process. Furthermore, process mining software allows inspecting individual traces and the detailed analysis of process variants.

Besides the mere discovery of a process model based on execution data (with a rather descriptive focus), process mining provides the ability to check whether a (possibly hand-made) process model matches the real-world execution and, if not, where the deviations lie. This type of process mining is subsumed under *conformance checking*.

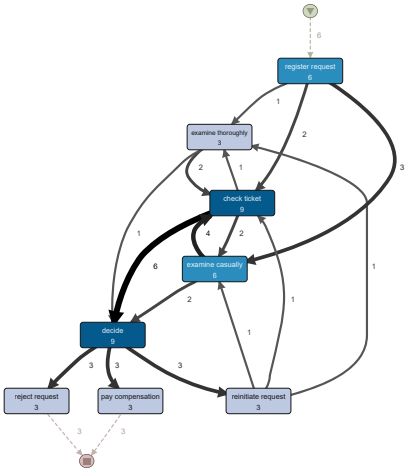


Figure 3.11: Discovery of Compensation Request Process in Disco [4] (100% Paths).

3.5.4 Conformance Checking

Conformance checking is especially suited to evaluate the *fitness* of process discovery algorithms [Wil16]. An event log is replayed on the discovered process model through, for instance, a token replay as it is known from Petri nets. This type of process mining can either detect real-world violations of normative process models or create suggestions for changing descriptive ones.

Rozinat and van der Aalst [RVdA08] have contributed one of the first approaches for conformance checking. Their technique is based on a fitness indicator measured by replaying the log. A behavioral and structural appropriateness criterion indicates whether the model allows for behavior not present in the log or contains unnecessary control structures and

duplicate activities, complicating the understanding of the process model. All previous elaborations are based on procedural process models. However, process mining can also be applied using the declarative modeling paradigm.

3.5.5 Declarative Process Mining

Researchers have also developed process discovery and conformance checking approaches for the declarative modeling paradigm. Generally, all types of process mining are also applicable for the declarative paradigm. More detailed elaborations and discussions of such approaches follow in Chapter 7. In general, it may be said that many approaches include a candidate set generation approach where an algorithm checks whether a potential set of constraints holds on the event log. Some techniques only support generating a subset of the Declare templates set. One challenge for declarative process mining is that with event logs getting larger, individual constraints may have less support and confidence as there is often no optimal solution. For imperative mining, this problem can be circumvented by showing more or fewer activities or paths of the process.

Maggi et al. [MBvdA12] and Kala et al. [KMDCDF16] contribute declarative discovery algorithms applying association rule and sequential pattern mining techniques alone or in combination on event logs, inspiring the approach of this thesis. The following provides an overview of applying process mining in practice.

3.5.6 Application in Practice

Process mining originates from academia, with Wil van der Aalst and others being its main driver and pioneer. Van der Aalst et al. [vdAAdM+12] have proposed the L* life-cycle model as a first reference point for practical

process mining projects; others have developed alternative methodologies and guidelines. Some are especially suited for a specific branch, e.g., healthcare. Researchers who are often inspired or influenced by success stories in practice develop new reference processes or methodologies.

Both practitioners and researchers create collections of case studies that describe the (successful) application of process mining. For instance, *HSPI S.p.A.*, an Italian consulting company, regularly creates a comprehensive collection of process mining applications during one year [11]. This "Database of Applications" serves as a good starting point when searching for process mining applications in particular branches or sectors. Banking & Insurance, Public Administration, and Manufacturing are just a few examples for them.

Reinkemeyer [Rei20] contributes an update about the current status of process mining in practice. First, he provides an overview of the principles and foundations of process mining. Then, case studies of successful process mining implementations in companies like Bosch, Siemens, or BMW deliver an understanding of the challenges and benefits of this technology in practice. The work concludes with an outlook on process mining's future academic and business challenges. It represents one of the first contributions that explicitly focuses on process mining in practice and brings back observations in the industry back to research.

Reference Models and Processes

In general, reference models are prototypical, comprehensive representations of concepts and serve as blueprints for constructing concrete instances of something. Such models can be called references processes whenever they depict a specific course of action that may or should be default for the described context. Typically, reference processes are applied in reality by instantiating the model elements and omitting certain

irrelevant or non-fitting parts for a particular scenario. For the process mining context, reference processes could help practitioners implement process mining in the best possible way.

One example for such a guideline is the work of Rebuge and Ferreira [RF12] that proposes a methodology for applying process mining in healthcare environments. Process mining projects, in general, consist of a planning phase where the goals and requirements are defined. Afterwards, the event data must be extracted from the relevant sources and integrated into an event log. The actual analysis follows, where employees involved in the project draw conclusions and develop results. Finally, the results are mapped to concrete actions which are put into practice. There are iterative cycles between each step to acknowledge new findings during the project. All in all, every process mining initiative follows more or less this strategy. However, the concrete manifestations of each step remain open and can be refined by project descriptions or methodologies.

L* Life-cycle Model Van der Aalst et al. [vdAAdM+12] have contributed the first reference model for process mining projects. Figure 3.12 shows an overview of the conceptual level. A process mining project consists of five stages. In stage 0, the general course of the project is planned, and the reasons for the application of process mining are clearly stated. Stage 1 encompasses extracting the event data, creating hand-made process models, and setting the objectives and questions precisely. Stage 2 and 3 include the actual process analysis using the three types of process mining (discovery, conformance checking, enhancement), striving for answers to the questions, and determining the concrete values of the KPIs. At last, in stage 4, process models and data extractions continuously monitor current process executions.

Later, van der Aalst extended this to the concept of "business process hygiene" where it is a standard procedure to apply process mining

to any business process [Wil16]. The results of stages 2 and 3 have to be interpreted concerning the questions within the project. These results can lead to interventions, adjustments, and redesigns of the objectives [vdAAdM+12].

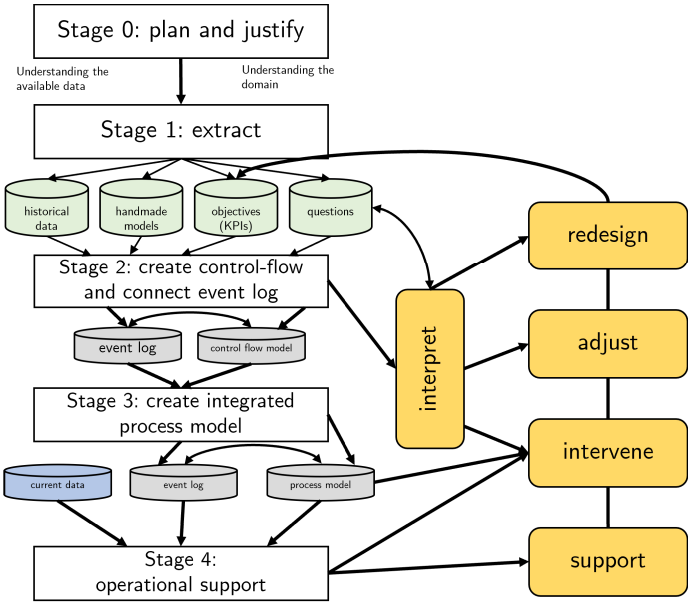


Figure 3.12: The L* Life-cyle Model for Process Mining Projects (Source: based on [vdAAdM+12]).

PM² The *PM²* process mining project methodology [VELLVDA15] can be seen as a response to the criticism on existing methodologies. Van Eck et al. claim that *PM²* addresses, for instance, the issue of the *Process Diagnostics Method (PDM)* [BGvdW09] that it is only tailored towards

smaller projects. The authors criticize both the *L** life-cycle model and *PDM* for not allowing explicit iterative analyses. According to them, L* is only suitable for discovering and integrating single, very structured processes.

Figure 3.13 shows the PM^2 process mining project methodology. It consists of six steps, clearly indicates input and output objects of each of them, and assigns roles (business experts, process experts). During the planning phase, goals are translated into research questions that shall be answered in the mining process. The main phase of the methodology where the extracted event data is processed, the actual mining and analysis, and an evaluation happen explicitly includes iteration cycles. Furthermore, it is possible to go back from evaluation to mining and data processing when refined or new research questions indicate this.

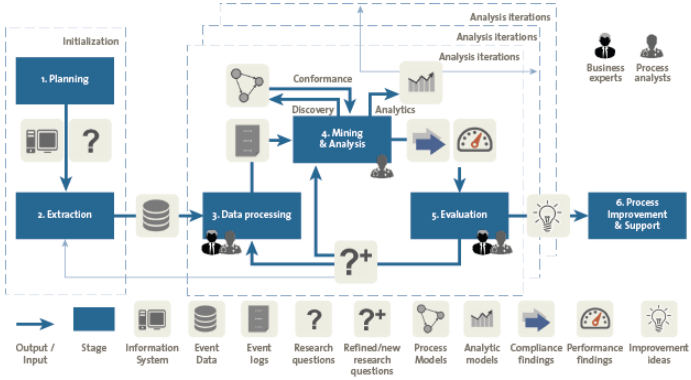


Figure 3.13: The PM^2 Process Mining Project Methodology (Source: [VELLVDA15, 26]).

Process Mining Project Methodology (ProMiPM) *ProMiPM* is based on research of Grohmann and Vossen. It is inspired by the well-known Business Model Canvas, whose foundations were laid by Osterwalder in 2004 [Ost04], an investment value canvas for data science projects [12] and methodologies related to process management such as the Horus method [SVOK12]. Several ideas of these concepts are combined into its canvas-like setup and appearance depicted in Figure 3.14. Although it has strong similarities with a canvas, it is still a methodology because it indicates a direction and consists of three main phases.

In [Rei20], van der Aalst claims that one of the current challenges for process mining is to bridge the gap between process mining and modeling. A general methodology can serve people with a background in normative process models and those who promote the use of process mining as a new approach for gaining As-Is models at the same time. Although process mining is successfully applied in practice, significant challenges remain. There is resistance whenever there is change, and process mining is no exception. ProMiPM supports overcoming this challenge in an organization by providing a global overview of a process for all stakeholders. In that sense, it represents a communication tool and is general enough to apply to any field of application, branch, or size of an organization.

At the same time, it is not only touching the highest abstraction level, scratching on the surface. Its goal is to concretely support a process mining project with an analysis of its characteristics, valuable for all process stakeholders. *ProMiPM* consists of 16 fields or steps, each of which addresses a different aspect relevant for process mining and has three phases. Fields in the pre-evaluation phase (Phase 1) are filled out when preparing the project, i.e., before the actual process mining starts. On the other hand, fields of the post-evaluation phase (Phase 3) are only filled out with assumptions or ideas beforehand. When the actual process mining is over, they are used to document the results.

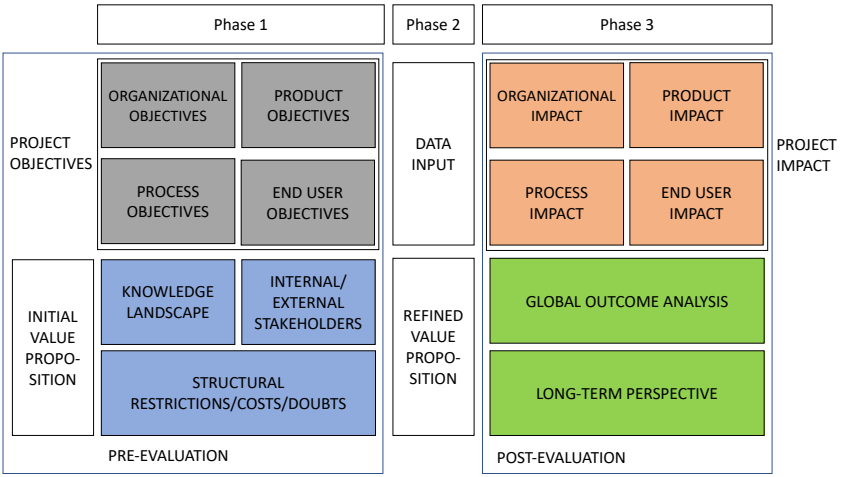


Figure 3.14: The Process Mining Project Methodology (ProMiPM).

Phase 2 encompasses the mining process with a refined value proposition and data input. The four fields with a gray background constitute the project objectives. All objectives together form the purpose of a project according to the definition in Reinkemeyer's work [Rei20]. Similarly, organizational, product, process, and end user impacts constitute the project outcome or impact. *ProMiPM*'s internal and external factor analysis has a blue, and the global post-analysis has a green background.

Further details and descriptions about the values of the fields are omitted here because the general overview is sufficient for sketching out a selection of reference models for process mining projects. Note that the questions and statements are not complete and only provide an impression of the content for each field. Project teams that apply the methodology for their process can quickly adapt the questions or add information suitable for their particular domain whenever they think this is appropriate.

After having introduced the essential terms and concepts for BPM and process mining, Chapter 4 continues with introducing two fundamental data mining techniques for pattern discovery, namely association rule and sequential pattern mining. Later in this thesis, process and data mining techniques are combined into a declarative process discovery approach.

4 Data Mining Techniques for Pattern Discovery

Data Mining denotes techniques and approaches that focus on extracting information and patterns in possibly large amounts of data. Leskovec et al. [LRU20] define data mining as "the discovery of models for data" whereby a model can, for instance, be a statistical, machine learning, or computational one. This thesis aims to apply data mining techniques to event data to perform process mining analysis. Since event logs are potentially massive datasets, combining both worlds seems to have promising benefits for understanding real-world processes.

One of the most prominent and widely-used techniques is association rule mining or learning, for which Agrawal et al. [AIS93] laid the foundations in 1993. Today, their approach is known as the Apriori algorithm. The goal is to discover significant patterns, relationships, and rules (between items) in a dataset consisting of transactions. A widespread application example is the market basket analysis, where marketing analysts want to find out which products customers commonly buy together. With that information, products that are likely to be bought together can be placed close by each other.

Another type of analysis that considers the temporal orders and relations between items is sequential pattern mining, also known as sequence mining. Sequential rule mining denotes the case where found patterns are translated to rules. Like the Apriori algorithm, sequential pattern mining originates from contributions of Agrawal et al. [AIS93]. In 1995,

Agrawal and Srikant [AS95] presented three algorithms for generating sequential patterns from a customer transaction database. Fournier-Viger et al. [FVLR+17] present a survey about a variety of sequential pattern mining techniques. Two representatives of this set of approaches are the *GSP* [SA96] and *SPADE* [Zak01] algorithms.

Case ID	Event ID	dd-MM-yyyy:HH.mm	Activity
1	35654423	30-12-2010:11.02	register request
1	35654424	31-12-2010:10.06	examine thoroughly
1	35654425	05-01-2011:15.12	check ticket
1	35654426	06-01-2011:11.18	decide
1	35654427	07-01-2011:14.24	reject request
2	35654483	30-12-2010:11.32	register request
2	35654485	30-12-2010:12.12	check ticket
2	35654487	30-12-2010:14.16	examine casually
2	35654488	05-01-2011:11.22	decide
2	35654489	08-01-2011:12.05	pay compensation
3	35654521	30-12-2010:14.32	register request
3	35654522	30-12-2010:15.06	examine casually
3	35654524	30-12-2010:16.34	check ticket
3	35654525	06-01-2011:09.18	decide
3	35654526	06-01-2011:12.18	reinitiate request
3	35654527	06-01-2011:13.06	examine thoroughly
3	35654530	08-01-2011:11.43	check ticket
3	35654531	09-01-2011:09.55	decide
3	35654533	15-01-2011:10.45	pay compensation

Figure 4.1: Three Cases of Insurance Compensation Request Process.

The motivation for applying these techniques to event logs in the context of process mining is as follows: Consider the first nineteen lines of the insurance compensation request process event log shown in Figure 4.1 representing three instances of the process. An association rule $register\ request \rightarrow decide$ or a sequential pattern *{register request, decide}* holds for all three cases and thus has a support of one. At the same time, a rule $register\ request \rightarrow pay\ compensation$ or pattern *{register request, pay compensation}* holds only in two out of three cases leading to support of $\frac{2}{3}$.

Overall, the idea for applying such fundamental data mining techniques to event logs is to take the activities of the process instances as the items or products of the famous market basket analysis example. Thereby, one case of the process corresponds to one customer buying the various products. With that in mind, analyses about activity coherences can be made that deliver insights into the actual process executions.

The following introduces association rule and sequential pattern mining more generally and in more detail whereby the focus is put on *FP-Growth* [HPY00] for generating frequent itemsets and the *GSP* [SA96] algorithm for discovering sequential patterns.

Table 4.1: Customer Transactions in a Computer Store (Source: based on [13]).

Transaction	**Items**
t1	{PC, Monitor, Mouse, Keyboard}
t2	{PC, Monitor, Keyboard}
t3	{PC, Monitor}
t4	{Monitor, Mouse, Keyboard}
t5	{Monitor, Mouse}
t6	{Mouse, Keyboard}
t7	{Monitor, Keyboard}

Let n be the number of transactions and X an itemset. Then, the support of X with respect to a set of transactions t_1, ..., t_n is defined as follows:

$$Support(X) = \text{Number of appearances of X in transactions } t_1, ..., t_n$$

Consider Table 4.1, containing customer transactions of a computer store (t_1 to t_7). The support value of itemset *{PC, Monitor}* is three be-

cause it is a subset of three of the seven transactions. Association rules create relations between at least two items. Semantically, they indicate dependencies between the existence of a set of items and the existence of another set. The support of an association rule $X \rightarrow Y$ is defined as the support of itemset X united with Y divided by the number of transactions. Content-wise, the support value indicates the share of transaction on which the association rule holds. The support value of the association rule $Monitor \rightarrow PC$ is $\frac{3}{7}$ because the support of itemset *{PC, Monitor}* is three and must be divided by seven (number of transactions).

$$Support(X \rightarrow Y) = \frac{Support(X \cup Y)}{N}$$

Another interest measure for association rules is confidence. It is defined as the support of itemset X united with itemset Y divided by the support of X. The confidence of X shows in how many of the cases where the items of itemset X are present, the items of Y are also present. For the transactions in Table 4.1, the confidence of the rule $Monitor \rightarrow PC$ is $\frac{3*7}{7*6} = \frac{1}{2}$ because the support of $Monitor \rightarrow PC$ is $\frac{3}{7}$ which must be divided by $\frac{6}{7}$ as the itemset *{Monitor}* appears in six out of the seven transactions, meaning that in half of the cases where customers bought a monitor they also bought a PC. Note that there the fractional support representation for an itemset has to be used.

$$Confidence(X \rightarrow Y) = \frac{Support(X \cup Y)}{Support(X)}$$

For both support and confidence, usually higher values are desired to discover representative and meaningful association rules on the dataset. Together, they create an impression of the significance of an association rule. However, there can be cases where the confidence value is misleading. In case the confidence of a rule $X \rightarrow Y$ is close to the support of the

consequent, the relation between X and Y may not be meaningful because the existence of Y may in reality not be influenced by the existence of X [14]. Therefore, alternative measurements like lift have been introduced over time. Lift is defined as follows:

$$Lift(X \rightarrow Y) = \frac{Confidence(X \rightarrow Y)}{\frac{Support(Y)}{N}}$$

If the lift of a rule is (significantly) higher than one the existence of itemset, X positively influences the existence of itemset Y.. They are independent if the lift value is close to one and negatively correlated if it is lower than one. Thus, the lift measure is another contribution for evaluating a rule's importance regarding the whole transaction set. The lift value of the rule $Monitor \rightarrow PC$ is $\frac{7}{6}$ because $\frac{\frac{1}{2}}{\frac{3}{7}} = \frac{7}{6}$. This means that buying a monitor is, in fact, relatively independent of purchasing a PC.

All in all, the lift measurement delivers another indicator of whether an association rule is "interesting" and users should take a closer look at it or just represents a statistical accumulation [14]. Other measures like conviction [BMUT97] or leverage [Pia91] do most likely not provide any additional benefit for evaluating their *interestingness* as the statistical independence aspect is already covered by lift.

4.1 Association Rule Mining

Association rule mining denotes the process of generating association rules based on a set of transactions. Each of the resulting rules relates a single item or set of items to another single item or set of items. Table 4.1 shows a sample set of market baskets found at the checkout in a computer store. The goal is to find patterns between products that are commonly

bought together. In that way, statements in the form "whenever someone buys product A there is a certain likelihood X that she or he also buys product B" can be constructed.

For all standard approaches, the critical part is to generate a set of frequent itemsets from which finally association rules can be generated. The following introduces *Apriori* as the oldest and most common technique. *FP-Growth* is an advanced, more efficient algorithm that relies on a tree-based structure to derive the frequent itemsets. Frequent itemsets require a minimum support value, whereas association rules can be evaluated with support, confidence, and lift.

4.1.1 Apriori

Generally speaking, the Apriori algorithm consists of three main steps [15]. It requires a minimum support threshold as input:

- Step 1: For each itemset of size one, calculate the support value.
- Step 2: Calculate the support value of the itemsets, remove those that do not fulfill the minimum required support value.
- Step 3: Based on the itemsets of the previous step kept, create itemsets with one item more and switch to step two with those. Stop as soon as no new itemsets that fulfill the support value can be constructed.
- (Step 4: Generate association rules based on the frequent patterns found).

The Apriori algorithm is based on the (anti-)monotonicity principle that subsets of frequent itemsets are also frequent and supersets of infrequent

itemsets are also infrequent, i.e., they do not fulfill the minimum support condition.

Figure 4.2 shows an application of the Apriori algorithm for the itemsets in Table 4.1 Using a minimum support of 0.5, i.e., the itemsets have to appear in at least four of the transactions, resulting in the following output: *{Monitor}*, *{Mouse}*, *{Keyboard}*, and *{Monitor, Keyboard}*. It uses twoiterations. In the first iteration, the algorithm constructs all itemsets of size one. The *{PC}* item does not fulfill the minimum support condition of 5; hence, it is not included in the set of frequent itemsets. Two of the three itemsets of size two do not fulfill the support condition. Therefore, only the itemset *{Monitor, Keyboard}* is kept. No further itemset *{Monitor, Mouse, Keyboard}* of size three is constructed because of the Apriori principle. The subsets *{Monitor, Mouse}* and *{Mouse, Keyboard}* of such an itemset were already found to be infrequent, i.e., they do not fulfill the minimum support requirement. Finally, the right side of Figure 4.2 shows all frequent itemsets.

While being an easy-to-understand algorithm, Apriori suffers from scalability and time complexity problems. Let d be the number of different products in the data set. Even though the Apriori principle allows a reduction of the item sets to be considered, the general functioning of the algorithm leads to 2^d item subsets requiring the database to be scanned d times to construct more candidate sets. This does not scale well for larger databases despite each scan being of linear complexity.

Therefore, the run time of Apriori is not so much dependent on the number of transactions but rather on the number of different products bought. For instance, another item *{Webcam}* in at least one of the transactions in Table 4.1 would enormously increase the time complexity because it had to be considered in every frequent itemset generation iteration. Tan et al. [TSK16] discuss the factors influencing the computational and time complexity. In the worst case, every candidate set of length k-1 must be

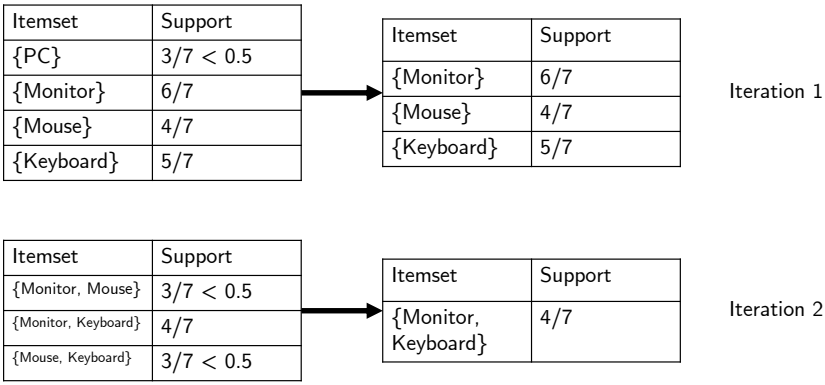


Figure 4.2: The Apriori Algorithm for Transactions in Table 4.1 (Minimum Support 0.5).

merged with another set of length k-1. This is of linear complexity but still does not scale for larger datasets. For this reason, researchers have developed alternative, more efficient algorithms for frequent itemset generation. One representative of these algorithms is *FP-Growth* [HPYM04].

4.1.2 FP-Growth

Han et al. [HPY00] claim that the popular FP-Growth algorithm delivers massive efficiency improvements compared to the Apriori algorithm. In contrast to Apriori-based approaches, it finds frequent itemsets without generating candidate sets. The basic idea is to construct a frequent-pattern tree (FP-tree) from which finally the frequent itemsets can be derived [TSK16]. FP-Growth performs the tree construction as follows: The items are sorted according to their appearances in the transaction. Items that do not fulfill the minimum support condition are discarded.

Then, starting from a root node (named null) for each transaction, paths with every item are constructed whereby a frequency counter is initialized with the value one. If a transaction shares a common prefix on an already existing path, the rest of the transaction is attached to the common prefix in the tree. The frequency counters for the common items are incremented by one.

Applying FP-Growth on the transactions in Table 4.1 leads to the progress shown in Figures 4.3 and 4.4. Assuming a minimum support requirement of 0.5, in the first step the *{PC}* item is removed as it only has a support of 3/7. The remaining products are ordered regarding their support in descending order (Monitor=6, Keyboard=5, Mouse=4). The transaction t_1 produces the first path of the FP-tree (Monitor, Keyboard, Mouse) with a frequency count of one. Since the ordered items for transactions t_1 to t_5 all share the full common prefix, no new paths or nodes are introduced to the tree, but only the frequency counts are incremented.

Transaction t_5 (*{Monitor, Mouse}*) shares a common prefix *{Monitor}* but skips the *Keyboard* item. Thus, a new node is attached to the *Monitor* node. Transaction t_6 does not share a common prefix. For this reason, a new path is introduced and attached to the *null* node. At last, transaction t_7 shares a common prefix again. Dotted lines between nodes representing the same items are required for constructing the frequent itemsets in the end [TSK16].

The process of deriving frequent itemsets from the FP-tree is illustrated in Figure 4.5, showing all paths containing the items *{Keyboard}* (left tree) and *{Monitor, Keyboard}* (right tree) based on the final FP-tree in Figure 4.4. First, the paths only containing single items are constructed. These sets of paths are called prefix-paths, for instance, of item *{Keyboard}* with a support of four. To evaluate whether, for instance, the item *{Monitor, Keyboard}* is frequent, first its prefix-path has to be constructed, which is shown in the right tree of Figure 4.5. The support value of *{Monitor,*

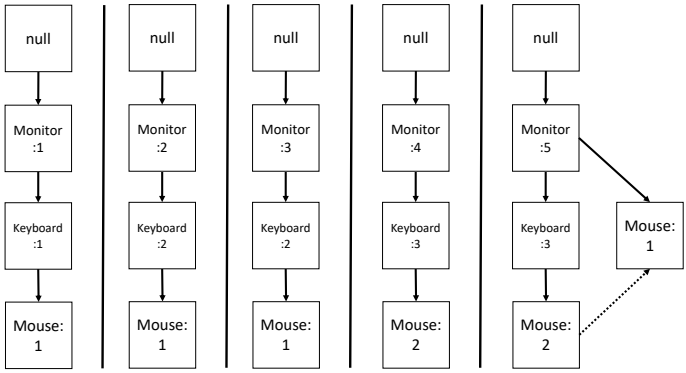


Figure 4.3: The FP-Growth Algorithm for Transactions in Table 4.1 (T_1 to T_5).

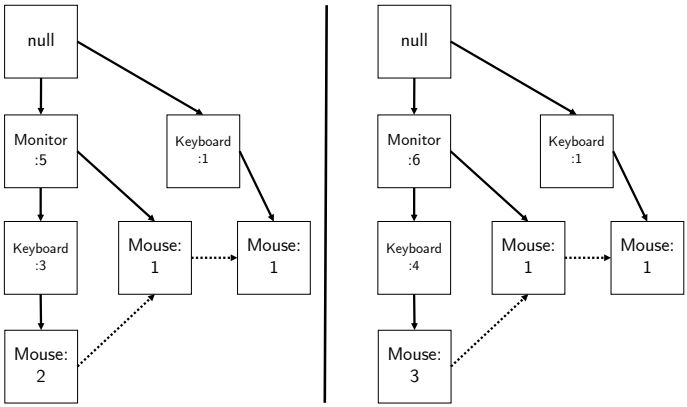


Figure 4.4: The FP-Growth Algorithm for Transactions in Table 4.1 (T_6 to T_7).

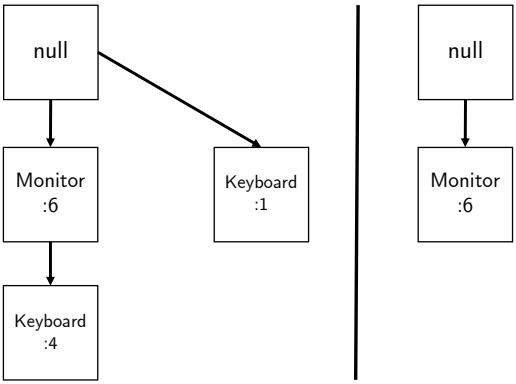


Figure 4.5: Prefix paths of {Keyboard} and {Monitor, Keyboard} (based on Last FP-tree in Figure 4.4).

Keyboard} can directly be read from this tree by adding up the leaves' values (four plus one equals five).

FP-Growth also divides the problem of finding frequent itemsets into subproblems. Whenever an itemset is found to be infrequent, the algorithm does not extend it but continues with others [TSK16]. Kosters et al. [KPP03] investigate the complexity of FP-Growth and find that it has performance advantages for cases with more different transaction items.

All in all, *FP-Growth* is an efficient algorithm that does not rely on generating candidate sets by extending frequent itemsets. Instead, it directly constructs FP-trees from the transactions database and derives the frequent patterns. Note that both *Apriori* and *FP-Growth* find the same set of frequent itemsets because there is only one correct solution assuming equal minimum support values are given.

After finding the frequent itemsets, they can generate association rules. Association rules consist of an antecedent and a consequent part, whereby

each consists of one or more items. Generating such rules requires itemsets of at least a length of two because no rule can be constructed from a single item. Typical are sizes of two to four items leading to rules with one to three elements in antecedent or consequent.

4.1.3 Association Rule Generation

In the most straightforward approach, all combinations of possible rules are constructed and evaluated whether they fulfill minimum confidence or other measure requirements (e.g., lift). For the frequent patterns resulting from the transaction in Table 4.1, there are only two possible association rules. The itemsets *{Monitor}*, *{Keyboard}*, and *{Mouse}* are single items and cannot be translated to any association rule. Itemset *{Monitor, Keyboard}* can be transformed to two association rules: $Monitor \rightarrow Keyboard$ and $Keyboard \rightarrow Monitor$. The former rule has a confidence value of $\frac{4}{6} = \frac{2}{3}$ while the latter has a confidence of $\frac{4}{5}$. Assuming a minimum confidence of $\frac{2}{3}$, both rules would be included in the final set of association rules. The support values of the rules are equal to the support values of the frequent itemsets they are based on.

Agrawal and Srikant [AS$^+$94] contribute an improved algorithm for generating association rules with more than one item in the consequent part. Its basic idea is that if rules of the form $ABC \rightarrow D$ do not meet the minimum confidence requirement, the rule $AB \rightarrow CD$ will not as well and, therefore, does not have to be further considered. The reason is the definition of confidence. All rules involving a set of items have the same numerator value. By removing an item from a set, its support can only increase or stay the same. Therefore, the term's denominator gets larger (or stays the same), and the overall confidence gets smaller (or stays the same). Hence, the rules constructed by shifting items from the antecedent to the consequent will also not meet the confidence requirement.

Association rules define relationships between items but can only find relations between the existence of items without considering their temporal order. Sequential pattern mining overcomes this shortcoming.

4.2 Sequential Pattern Mining

Sequential pattern mining was first introduced by Agrawal and Srikant [AS95] in 1995. They propose three algorithms, whereby two of them are based on the same candidate generation principle as the Apriori algorithm for association analysis. According to the definition of Agrawal and Srikant [AS95], sequential pattern mining requires two pieces of additional information, namely information about time with either concrete timestamps or at least a temporal order of the items and an assignment of the transactions to a customer.

Items in such sequences do not need to follow each other directly in reality, but there can be items in between that are not part of a specific sequence. In contrast, episode mining aims for discovering frequent sequence patterns with items that are "close to each other" [MTIV97]. Table 4.2 shows a transaction database with items bought by customers in a computer store. Table 4.3 presents the same data with the entire customer sequence assigned to a customer in one cell.

Note that the elements of the sequences can consist of more than one item. The support of a sequence X with elements $i_{1, ..., n}$ is defined similar to association rule mining as the frequency in which a sequence is subsequence of a transaction. For instance, the sequence {Keyboard} has a support of $\frac{4}{5}$ as it appears in four of the five transactions. Setting the minimum support the 0.25, the frequent sequential patterns would be the following: {Keyboard}, {Mouse}, {GPU}, {Webcam}, {Keyboard, Mouse}, {Keyboard, GPU}, {Mouse, GPU}, {Keyboard, Webcam}, and {Keyboard, {Mouse, GPU}}. Some algorithms like *AprioriSome* [AS95] with standard

Table 4.2: Computer Store Transaction Data (Source: based on [AS95]).

Transaction Time	**Customer ID**	Items Bought
August 10 2021	2	PC, Monitor
August 12 2021	5	Webcam
August 15 2021	2	Keyboard
August 20 2021	2	Mouse, SSD, GPU
August 25 2021	4	Keyboard
August 25 2021	3	Keyboard, Software, GPU
August 25 2021	1	Keyboard
August 30 2021	1	Webcam
August 30 2021	4	Mouse, GPU
September 25 2021	4	Webcam

Table 4.3: Computer Store Transactions per Customer (Source: based on [AS95]).

Customer ID	**Customer Sequence**
1	{{Keyboard}, {Webcam}}
2	{{PC, Monitor}, {Keyboard}, {Mouse, SSD, GPU}}
3	{{Keyboard, Software, GPU}}
4	{{Keyboard}, {Mouse, GPU}, {Webcam}}
5	{{Webcam}}

settings only produce sequences with maximum length. In this case, only {Keyboard, Webcam}, and {Keyboard, {Mouse, GPU}} remain. Srikant and Agrawal [SA96] contribute improvements for their approach. Today, this algorithm is known as the Generalized Sequential Pattern (GSP) pattern

algorithm. The following introduces its basic operation and applies it on a sample data snippet.

4.2.1 Generalized Sequential Pattern Mining (GSP)

The GSP algorithm is a direct successor of the three algorithms by Srikant and Agrawal [AS95]. They identify three shortcomings of their previous contributions, focusing on the sequential pattern mining problem description. First, users cannot specify a time window for items' minimum and maximum time distance in a frequent sequential pattern. In the business process context, such a feature could be convenient to focus on patterns that are more close to each other. Consider a process where one set of activities happens a few hours and another one days or weeks afterwards. In this case, patterns between activities of the two sets may not be meaningful.

Moreover, users cannot define a time window in which several items elements are treated as a single element. This could be useful in cases where some elements form a group of elements belonging together that should be treated as such in the resulting sequential patterns. Consider a set of events belonging to a superordinate task of the process. e.g., five manufacturing steps for the assembly of a table. Process modelers would most likely represent them in a subprocess. If users know the time period in which the steps usually take place, they can define a sliding time window so that the GSP algorithm handles them as one activity.

Another advantage of the GSP algorithm in comparison to the algorithms in [AS95] is that users can specify taxonomies of elements. This means that generalizing and specializing items can replace elements in resulting sequential patterns. For instance, the *Cologne cathedral* is one representative of the *Buildings in Cologne* category and a *software application* is a special *application*. This feature of the GSP algorithm could

Table 4.4: Computer Store Transactions with Time Differences (Source: based on [SA96]).

Customer ID	Transaction Time	Items Bought
1	1	Keyboard
1	2	Webcam
2	1	PC, Monitor
2	20	Keyboard
2	50	Mouse, SSD, GPU
3	1	Keyboard, Software, GPU
4	5	Keyboard
4	50	Mouse, GPU
4	60	Webcam
5	15	Webcam

be exploited similarly to the sliding time window parameter. Consider a taxonomy that assigns five manufacturing steps to a *manufacturing* item. The superordinate concept could automatically replace concrete activities in sequential patterns.

If the minimum time distance is zero, the maximum time distance is infinite and the sliding time window to differentiate elements is also zero, the GSP algorithm produces the same results as the algorithms introduced in the previous contribution [AS95]. As usual, the goal is to find all frequent sequences that fulfill the requirement of a minimum user-specified support threshold. The GSP algorithm fits process mining activities because of the three additional parameter options. Besides, it is performant and can be applied to larger datasets [SA96]. However, also other algorithms like SPADE [Zak01] could be applied to event logs without any problems.

Table 4.4 shows the same transaction data as in Table 4.2 with transactions sorted by customer ID and different timestamp information in numerical format. If no minimum and maximum gap and no sliding windows is defined the GSP algorithm finds the same frequent sequences as the other algorithms: {Keyboard}, {Mouse}, {GPU}, {Webcam}, {Keyboard, Mouse}, {Keyboard, GPU}, {Mouse, GPU}, {Keyboard, Webcam}, and {Keyboard, {Mouse, GPU}} for a minimum support of 0.25.

With a sliding window of five, the sequence {Keyboard, Webcam} would not be included in the set of frequent sequences anymore. The reason is that items {Keyboard} and {Webcam} are treated as one single element because their transaction time difference is only one. Setting a maximum gap of 40 leads to a set of frequent sequences where the sequence {Keyboard, {Mouse, GPU}} is not included anymore since the transaction time difference of {Keyboard} and {Mouse, GPU} for customer id four is 45. Therefore, this sequence is only found for customer id two and no longer supports the minimum support requirement. Overall, the GSP algorithm provides additional freedom for users to limit and adapt the generation of sequential patterns. At the same time, it still allows the standard discovery approach when using the standard parameter settings. The basic procedure of the GSP algorithm is as follows:

- Step 1: Calculate the support value of all sequences of length one, i.e., single items.
- Step 2: Extend the frequent sequences with one more item and check whether it still fulfills the support requirement.
- Step 3: Drop the infrequent sequences.
- Step 4: Repeat step two and three until no additional sequences can be constructed.

Like the Apriori algorithm for frequent itemset generation [AS95], the GSP algorithm is based on the principle of candidate set generation. Potential sequential patterns are evaluated regarding whether they fulfill the minimum support requirement. Although the principle is the same, the GSP algorithm uses some more advanced mechanisms to generate the candidate sets and determine the support value of a sequential pattern.

That is because sequential pattern mining deals with sequences whose join mechanism is not as trivial as for itemsets without any order of the elements. Srikant and Agrawal [SA96] introduce the definition of a "*contiguous subsequence*". A sequence c is a contiguous subsequence of sequence s if either c only misses the first or last item of s or misses only one item of a multi-item element of c (subsequence with at least two items). Another possibility to characterize c as a contiguous subsequence is if it fulfills one of the two properties for another sequence that is a contiguous subsequence of s itself (transitive contiguousness).

The GSP algorithm starts with the single item sequences and joins them with each other to produce sequences of length two. Starting with single item sequences, they have to be constructed in two ways: In one join mechanism, the resulting sequence consists of an item with two elements, and in another one, they appear as two disjoint elements. Table 4.4 consists of the following sequences of length one: {Keyboard}, {Webcam}, {Mouse}, {GPU}, {Software}, {PC}, {Monitor}, and {SSD}. Table 4.5 shows the joint candidate sequences of length two leading to 8 x 14 = 112 sequences.

In the next step, all sequences with non-frequent contiguous subsequences are removed from the set of candidate sequences. For the sequences shown in Table 4.5 this leads to the remaining sequences of length two shown in Table 4.6. Setting a minimum support requirement of 0.25, all sequences that contain {Software}, PC, {Monitor}, and {SSD} are deleted from the candidate set because these items were in the market basket of only one of the five customers. Then, if the maximum gap pa-

Table 4.5: Joined Candidate Sequences of Length Two (Transactions of Table 4.4).

First Item	Candidate Sequence of Length Two
Keyboard	{{Keyboard}, {Webcam}}, {Keyboard, Webcam}, {{Keyboard}, {Mouse}}, {Keyboard, Mouse},{{Keyboard}, {GPU}}, {Keyboard, GPU}, {{Keyboard}, {Software}}, {Keyboard, Software}, {{Keyboard}, {PC}}, {Keyboard, PC}, {{Keyboard}, {Monitor}}, {Keyboard, Monitor}, {{Keyboard}, {SSD}}, {Keyboard, SSD}
Webcam	{{Webcam}, {Keyboard}}, {Webcam, Keyboard}, {{Webcam}, {Mouse}}, {Webcam, Mouse}, {{Webcam}, {GPU}}, {Webcam, GPU}, {{Webcam}, {Software}}, {Webcam, Software}, {{Webcam}, {PC}}, {Webcam, PC}, {Webcam, PC}, {{Webcam}, {Monitor}}, {Webcam, Monitor}, {{Webcam}, {SSD}}, {Webcam, SSD}
Mouse	{{Mouse}, {Keyboard}}, {Mouse, Keyboard}, {{Mouse}, {Webcam}}, {Mouse, Webcam}, {{Mouse}, {GPU}}, {Mouse, GPU}, {{Mouse}, {Software}}, {Mouse, Software}, {{Mouse}, {PC}}, {Mouse, PC}, {{Mouse}, {Monitor}}, {Mouse, Monitor}, {{Mouse}, {SSD}}, {Mouse, SSD}
GPU	{{GPU}, {Keyboard}}, {GPU, Keyboard}, {{GPU}, {Webcam}}, {GPU, Webcam}, {{GPU}, {Mouse}}, {GPU, Mouse}, {{GPU}, {Software}}, {GPU, Software}, {{GPU}, {PC}}, {GPU, PC}, {{GPU}, {Monitor}}, {GPU, Monitor}, {{GPU}, {SSD}}, {GPU, SSD}
Software	{{Software}, {Keyboard}}, {Software, Keyboard}, {{Software}, {Webcam}}, {Software, Webcam}, {{Software}, {Mouse}}, {Software, Mouse}, {{Software}, {GPU}}, {Software, GPU}, {{Software}, {PC}}, {Software, PC}, {{Software}, {Monitor}}, {Software, Monitor}, {{Software}, {SSD}}, {Software, SSD}
PC	{{PC}, {Keyboard}}, {PC, Keyboard}, {{PC}, {Webcam}}, {PC, Webcam}, {{PC}, {Mouse}}, {PC, Mouse}, {{PC}, {GPU}}, {PC, GPU}, {{PC}, {Software}}, {PC, Software}, {{PC}, {Monitor}}, {PC, Monitor}, {{PC}, {SSD}}, {PC, SSD}
Monitor	{{Monitor}, {Keyboard}}, {Monitor, Keyboard}, {{Monitor}, {Webcam}}, {Monitor, Webcam}, {{Monitor}, {Mouse}}, {Monitor, Mouse}, {{Monitor}, {GPU}}, {Monitor, GPU}, {{Monitor}, {Software}}, {Monitor, Software}, {{Monitor}, {PC}}, {Monitor, PC}, {{Monitor}, {SSD}}, {Monitor, SSD}
SSD	{{SSD}, {Keyboard}}, {SSD, Keyboard}, {{SSD}, {Webcam}}, {SSD, Webcam}, {{SSD}, {Mouse}}, {SSD, Mouse}, {{SSD}, {GPU}}, {SSD, GPU}, {{SSD}, {Software}}, {SSD, Software}, {{SSD}, {PC}}, {SSD, PC}, {{SSD}, {Monitor}}, {SSD, Monitor}

rameter is set to infinity, all other infrequent subsequences are removed. Table 4.7 shows the remaining sequences of length two included in the set of sequential patterns found by the GSP algorithm.

These sequences of length two join with each other if the following condition holds: Removing the first item of one sequence leads to the same result as dropping the last item of another sequence. The items can be single items or be part of an itemset. This procedure stops when no additional candidate sequences can be generated through this join mechanism. Taking the sequences in Table 4.7 {{Keyboard}, {Mouse}} and {Mouse, GPU} can be joined to {{Keyboard}, {Mouse, GPU}} because deleting the first and last item leads to the item {Mouse}. Other joins are not possible. The final sequence of length three is not removed from the candidate set because {{Keyboard}, {Mouse}, {{Keyboard}, {GPU}}, as well as {Mouse, GPU} are frequent (no infrequent contiguous subsequence). The GSP algorithm stops with that set of sequences because only a single item remains, and no further joins are possible. The final output of the algorithm consists of the sequences in Table 4.7 and 4.8 plus the four sequences with a single item.

Another unique aspect of the GSP algorithm is how it determines the support values of the sequences. For frequent itemsets without any sequential order aiming for discovering association rules, one pass over all transaction is enough to determine the support of an itemset that, for instance, includes three specific items. However, things are different for sequential pattern mining. For instance, the sequences {Keyboard, Mouse, GPU} and {Mouse, GPU, Keyboard} are not equivalent. Therefore, several independent passes over the data are required for determining the support count of each sequence that consists of the same elements. This is why a stronger focus has to be laid on the efficient calculation of support values for candidate sequences for which the GSP algorithm uses a special tree structure that consists of a mix of lists of sequences and hash tables.

Table 4.6: Joined Candidate Sequences of Length Two After Pruning (1/2).

First Item	Candidate Sequences of Length Two
Keyboard	{{Keyboard}, {Webcam}}, {Keyboard, Webcam}, {{Keyboard}, {Mouse}}, {Keyboard, Mouse}, {{Keyboard}, {GPU}}, {Keyboard, GPU}
Webcam	{{Webcam}, {Keyboard}}, {Webcam, Keyboard}, {{Webcam}, {Mouse}}, {Webcam, Mouse}, {{Webcam}, {GPU}}, {Webcam, GPU}
Mouse	{{Mouse}, {Keyboard}}, {Mouse, Keyboard}, {{Mouse}, {Webcam}}, {Mouse, Webcam}, {{Mouse}, {GPU}}, {Mouse, GPU}
GPU	{{GPU}, {Keyboard}}, {GPU, Keyboard}, {{GPU}, {Webcam}}, {GPU, Webcam}, {{GPU}, {Mouse}}, {GPU, Mouse}

Sequential patterns are sequences that fulfill a certain minimum support requirement providing a good overview of the sequential orders in customer transactions. Sometimes, however, it is desirable to represent these sequences as rules that create a relation between two sequences. The relationship between sequential patterns and sequential rules is analogue to the frequent itemsets and the association rules derived from them.

4.2.2 Sequential Rule Mining

Fournier-Viger et al. [FVGZT14] differentiate between two types of sequential rule mining: One type produces rules that have unordered sets of events in both antecedent and consequent parts. Their ERMiner approach or the RuleGrowth algorithm by Fournier-Viger et al. [FVNT11] generate

Table 4.7: Joined Candidate Sequences of Length Two After Pruning (2/2).

First Item	Candidate Sequences of Length Two
Keyboard	{{Keyboard}, {Webcam}} {{Keyboard}, {Mouse}}, {{Keyboard}, {GPU}}
Mouse	{Mouse, GPU}

Table 4.8: Joined Candidate Sequences of Length Three After Pruning.

First Item	Candidate Sequences of Length Two
Keyboard	{{Keyboard}, {Mouse, GPU}}

such output. For this thesis, they are not considered in larger detail. The reason is that the respective sets of events do not specify a particular order. For the application in the business process context later in this thesis, however, it is very desirable to know the order of process activities. Also, information would be lost when transforming a sequential pattern into a sequential rule.

Therefore, the second type of sequential rule mining includes sequences in the antecedent and consequent parts of the rule. Lo et al. [LKW09] present groundwork definitions for sequential rules and a mining algorithm that generates sequential rules based on sequential patterns found from the PrefixSpan algorithm by Pei et al. [PHMA+01]. Pham et al. [PLHV14] contribute an approach for constructing sequential rules based on patterns in a prefix tree. Fournier [16] shows the fundamentals of sequential rule mining and provides a set of open-source, java-based libraries for pattern discovery [17].

Two important terms here are closed sequential patterns and non-redundant sequential rules. Closed sequential patterns are patterns that are not included in any other pattern with the same support. They represent a minimal set of frequent sequences (sequential patterns) with which all other sequential patterns can be constructed again [PLHV14]. Their purpose is to reduce the number of patterns that have to be considered to construct sequential rules. For instance, in Table 4.7 and 4.8 only {{Keyboard},{Webcam}} and {{Keyboard}, {Mouse, GPU}} are closed because {{Keyboard},{Webcam}}, {{Keyboard}, {Mouse}} and {{Keyboard},{GPU}} as well as the frequent one-element sequences can be derived from them. A non-redundant sequential rule is a rule that cannot be inferred by another rule. Consider the sequential rules $Keyboard \rightarrow Mouse$ and $Keyboard \rightarrow Mouse, GPU$. The former is a redundant rule because it can be derived from the latter. Analogue to association rules, support and confidence are defined as follows:

$$Support(X \rightarrow Y) = \frac{Support(X + Y)}{N}$$

whereby X and Y are sequences and the plus-operator indicates a concatenation of two sequences. The following term calculates the confidence value of a sequential rule:

$$Confidence(X \rightarrow Y) = \frac{Support(X + Y)}{Support(X)}$$

Consider the rule $Keyboard \rightarrow Webcam$, derived from the closed sequential pattern {{Keyboard},{Webcam}} with a support of 2. The confidence is $\frac{2}{4} = 0.5$. For the rule $Keyboard \rightarrow Mouse, GPU$ the support is 2 and the confidence is $\frac{2}{4} = 0.5$ as well. Generally, sequential rules can be derived from any sequence of at least length two. Besides the confidence value, also other measures like lift can be determined.

After having introduced two fundamental data mining techniques for pattern and rule discovery, Chapter 5 continues with their application on process event data for a declarative process mining approach. The fundamental requirement is event data which can be represented in a format equivalent to the customer transaction data.

Part II

Combining Data and Process Mining

Part 2 ("Combining Data and Process Mining") of this thesis combines data and process mining. Chapter 5 shows how association rule and sequential pattern mining can be applied to event logs for process mining. The approach in this thesis uses a RapidMiner [6] script that loads the event log, performs several preprocessing steps, and finally executes the data mining techniques. All steps are performed on a small insurance compensation request process but are generally transferable to other event logs that fulfill the minimum requirements for process mining (case ID, activity name, timestamp). Next, the chapter demonstrates how different types of association rules and sequential patterns or rules (varying regarding, e.g., size or inclusion of additional attributes) can be transformed to constraints based on the (multi-perspective) Declare template set. Additionally, parameter settings like support, confidence, or inclusion and exclusion of attributes are mentioned. Furthermore, Chapter 5 discusses to what extent they influence the transformation process or the meaningfulness of the declarative process model.

Chapter 6 uses the RapidMiner script and applies it to larger sample event logs from web repositories, resulting in full declarative models and a discussion about the benefits of each application and a rule/pattern or Declare representation in general.

5 Applying Pattern and Rule Mining on Process Event Data

This chapter combines the techniques for pattern discovery with process mining and shows how to apply association rule and sequential pattern mining to event logs suitable for process mining. For this purpose, the event data has to be transformed and represented to be comparable to the transactions in the market baskets standard example. Transactions in this process context are instances (executions) of processes. An item is a single activity of the process, and sequences are successions of activities in a particular process. Overall, the goal is to produce association rules and sequential patterns (or rules) that uncover interesting, beneficial relations for users in process analysis environments. Finally, some results are transformed to model elements in the Declare notation to build a declarative process model.

5.1 General Setup of the Implementation

The approach in this thesis requires a standard event log suitable for process mining activities in tool environments like Celonis [5] or Disco [4]. It is implemented in RapidMiner [6], a tool for all kinds of activities in the data transformation and mining field which is Java-based and implements most of the functionalities in the prominent Weka tool [FHW16]. This thesis uses RapidMiner for all data mining activities on process event logs.

One of its advantages is that it provides the user with operators they can select and include into a RapidMiner process via a *Drag&Drop* mechanism. Examples for such operators are accesses to all kinds of file formats, attribute type changes, discretization, or model generation and applications. Assuming a sound process with matching inputs and outputs of the operators, users can execute the data process by clicking a button. RapidMiner supports data import through either CSV or Excel format.

Once imported into the application, users can load the data into a process with a single operator. The overall idea for the approach in this thesis is that users only have to change data input and can then reuse the remaining process without changing the remaining structure. However, they can adapt the operators' parameters. Users can vary support and confidence values for rule generation or include or exclude optional event data attributes. One disadvantage of RapidMiner in comparison to, for instance, Python scripts is that it does not provide the flexibility of code-based applications. The assumption is, though, that the clear set of operators provides the users an easy-to-use toolbox for their process analysis activities. Also, more advanced users can extend RapidMiner processes and embed Python scripts.

The following presents the RapidMiner processes for generating association rules and sequential patterns from event log input with a detailed inspection of each parameter's input, output, and parameters. Additionally, each process shows how the resulting patterns and rules should be interpreted and how they can be translated to a part of a declarative process model in the Declare notation. The outputs of the RapidMiner processes and the declarative process models should form a close connection with a fast translation between the resulting patterns or rules and the model elements and vice versa.

5.2 Association Rule Mining on Event Logs

Figure 5.1 presents a process for the application of association rule mining on an event log in the RapidMiner tool [6]. Before applying association rule mining on an event log, several preprocessing steps have to be performed. Assuming an event log in the usual form like in Figure 3.8, in a first step, the activities of a trace have to be grouped by the *Case ID*. This is done by the *Aggregate* operator with *Activity* as the aggregation attribute and *concatenation* as the aggregation function. For the compensation request process example, the output of the aggregation looks like in Table 5.1. Now that every row of the data represents one case, the activities are concatenated in one cell. The cells for the activities have to be split up. In this case, this happens by applying a split operation on the *Activities* column with "|" as the split pattern so that every activity has its own cell or column.

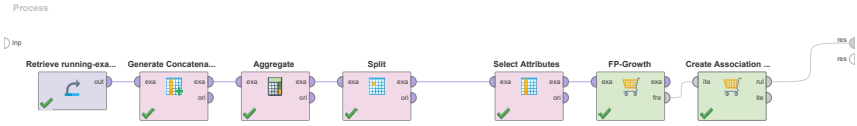


Figure 5.1: Process of Association Rule Mining on Event Logs in RapidMiner [6].

Then, only the activity columns are selected as an input for generating the frequent itemsets (operator *Select Attributes)*. The *FP-Growth* operator generates frequent itemsets of activities. For the process context, they mean activities that appear together in many executions, i.e., cases of the process. Users can set the minimal support or frequency of an itemset and the minimum and maximum number of items per itemset and number of itemsets. Starting with standard settings of a minimum support of 0.8,

Table 5.1: Compensation Request Log with Activities Aggregated by Case ID.

Case Id	Activities
1	register request | examine thoroughly | check ticket | decide | reject request
2	register request | check ticket | examine casually | decide | compensation
3	register request | examine casually | check ticket | decide | reinitiate request | examine thoroughly | check ticket | decide | pay compensation
4	register request | check ticket | examine thoroughly | decide | reject request
5	register request | examine casually | check ticket | decide | reinitiate request | check ticket | examine casually | decide | reinitiate request | examine casually | check ticket | decide | reject request
6	register request | examine casually | check ticket | decide | pay compensation

itemset sizes of two to three, and no maximum or minimum number of itemsets is advisable.

Table 5.2 shows frequent itemsets based on the following settings: Minimum support of 0.9, minimum items per itemset of 2, maximum items per itemset of 3, and maximum number of itemsets of 100. The minimum number of itemsets is 10. Although the minimum support is 0.9, the algorithm finds six itemsets with lower support because the minimum number of items was set to 10. Frequent itemsets are the input for generating association rules.

The *criterion* parameter of the *Create Association Rules* operator defines the criterion which is used for generating the rules, the standard setting is confidence, but also others like lift or gain are possible. Figure 5.3 presents an excerpt of the association rules found with a minimum confidence of 0.8. All rules have one or two items in their premise and conclusion parts, resulting from the frequent itemset generation operator for which the maximum number of itemsets was set to three.

Table 5.2: Frequent Itemsets of Compensation Request Process Event Log Activities.

Size	Support	Item 1	Item 2	Item 3
2	1.0	check ticket	decide	
2	1.0	check ticket	register request	
2	1.0	decide	register request	
3	1.0	check ticket	decide	register request
2	0.667	check ticket	examine casually	
2	0.667	decide	examine casually	
2	0.667	register request	examine casually	
3	0.667	check ticket	decide	examine casually
3	0.667	check ticket	register request	examine casually
3	0.667	decide	register request	examine casually

Table 5.3 shows association rules that only include activities of the compensation request process. However, it is also possible to use the other optional attributes in the event log. RapidMiner provides the *Generate Concatenation* operator to combine two attributes of the same row. The compensation request process event log contains resource information for every activity execution, indicating the employee who has executed one particular step. After concatenating *activity* and *resource* attribute, Table 5.1 is extended with another column that contains the same activities

Table 5.3: Association Rules for Compensation Request Process Event Log.

Premises	Conclusion	Support	Conf.
check ticket, decide	register request	1.0	1.0
check ticket, register request	decide	1.0	1.0
decide, register request	check ticket	1.0	1.0
check ticket	decide	1.0	1.0
decide	check ticket	1.0	1.0
check ticket	register request	1.0	1.0
register request	check ticket	1.0	1.0
decide	register request	1.0	1.0
register request	decide	1.0	1.0
check ticket	decide, register request	1.0	1.0
decide	check ticket, register request	1.0	1.0
register request	check ticket, decide	1.0	1.0
...	...	...	...

as before, where each of them is combined with the associated resource attribute. Afterwards, the RapidMiner process stays the same as before, with the only difference that both the columns with single activities and activity-resource combinations are provided as an input for frequent itemset and association rule generation. The attribute combinations alone may not lead to frequent itemsets with appropriate support values.

Using the insurance compensation request process with the same settings as before, RapidMiner produces 36 association rules. The reason is that the process finds many combinations of resource *Sara* and the *decide* activity and the other activities of the process. Therefore, Table 5.4 shows only an excerpt with five association rules. Line 1, for instance, can be

Table 5.4: Association Rules for Compensation Request Process Event Log with Resource Information.

Premises	Conclusion	Support	Conf.
check ticket	decide/Sara	1.0	1.0
check ticket, decide/Sara	decide	1.0	1.0
decide, decide/Sara	check ticket	1.0	1.0
check ticket, decide	register request	1.0	1.0
check ticket, register request	decide	1.0	1.0
...	...	...	...

interpreted in a way that every checked ticket is finally decided by *Sara*. Still, the process finds association rules that do not include any resource information (e.g., line 4 of Table 5.4). It is also possible to use continuous attributes in the association rule generation process. For this, continuous attributes (for instance, costs) must be discretized so that every value is mapped to a specific category. Then, users can perform the same process as with the resource attributes.

The process of generating declarative constraints from an event log is not over yet. In a final step, the association rules have to be translated to declarative process model elements. Altogether, the transformations form the set of declarative model elements and thereby the declarative process model.

Confidence is more important for selecting association rules for transformation. Constraints with high confidence values hold on the process in more cases and thus, provide a more precise declarative description. The number of times a constraint is active, but the actual process execution as indicated by the event log does not fulfill it, is lower. Users should most likely go for settings above 0.8. For lower values, active constraints

describe incorrect behavior for more than 20% of the cases, which, most likely, is not acceptable for most scenarios. Support, in this case, is more closely related to the relative importance of a constraint. Rules with higher support values create constraints that can be applied to more cases of the process, i.e., they are not vacuously satisfied and provide an overview of standard behavior of the process.

Chapter 6 addresses the proper selection of support and confidence values for frequent itemset and rule generation and the advantages and drawbacks that come with it in more detail. The following presents an approach for transforming association rules to model elements in Declare.

5.2.1 Translating Association Rules to Declare Constraints

In the simplest case, a rule consists of one activity as the premise and one as the conclusion. Users decide to include this rule in a declarative process model based on support and confidence. For the 1-to-1 relationship, one can translate the rule to a **RESPONDED EXISTENCE** template where the existence of the first activity enforces the existence of the second. For the rules in Table 5.3, line 4 with activities *check ticket* and *decide* form such a constraint, meaning that whenever a ticket is checked, there should also be a decision about it. A graphical representation of the constraints using the Declare notation could look as shown in Figure 5.2.

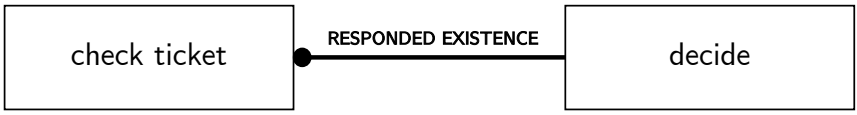


Figure 5.2: RESPONDED EXISTENCE Constraint for Two Activities.

Another type of setup is an association rule with one activity as the premise and two activities as the conclusion. The translation of such rules

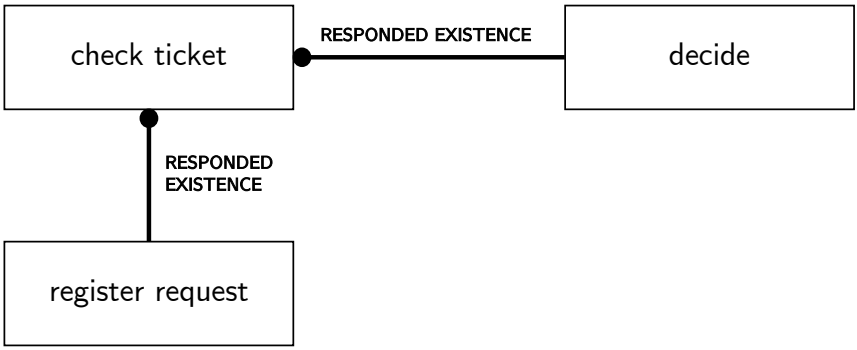


Figure 5.3: RESPONDED EXISTENCE Constraint for Three Activities (1 Premise, 2 Conclusions).

to Declare constraints can be done by splitting the rule into two independent **RESPONDED EXISTENCE** templates. Table 5.3 shows a rule with *check* as the premise and *decide* and *register request* as conclusions.

Figure 5.3 shows a graphical representation of both **RESPONDED EXISTENCE** constraints. The previous example represents a subset of the rule and translation here. So additional to the graphical depiction of the first constraint, the construct contains another line from the *check ticket* to the *register request* activity. Both constraints are independent of each other.

There is no direct translation to Declare if a rule has two activities as the premise and one as the conclusion. The standard set of the Declare notation does not include a **RESPONDED EXISTENCE** constraint with more than one activity on one side of the constraint. However, it is possible to extend the Declare notation by introducing new LTL formulas and adding own constraint templates. Figure 5.4 shows a **RESPONDED EXISTENCE** constraint for the *{check ticket, register request}* $\rightarrow$ *{decide}*

rule. This rule requires both premises to appear in the process and not only one of them.

However, the *{check ticket}* → *{decide}* relation still also holds. Rule generation has shown that the larger rule may be more expressive than the first one. It remains open how to explicitly include that the *{check ticket}* → *{decide} and {register request}* → *{decide}* rules also hold in the Declare representation. One option is to introduce a **RESPONDED EXISTENCE** label on both connections like depicted in Figure 5.4. If the confidence values of the rules with single-item antecedents are too low, there is only one label for both connections.

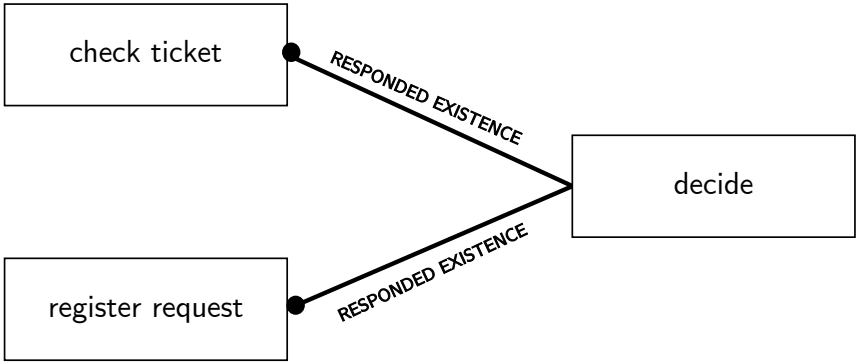


Figure 5.4: RESPONDED EXISTENCE Constraint for Three Activities (2 Premises, 1 Conclusion).

When setting the maximum number of items per itemset for frequent itemset generation to four, the rule generation operator also generates rules with four items. Using the same event data as before, the process produces a rule *{decide, examine casually}* → *{check ticket, register request}* (not included in Table 5.3). Figure 5.5 presents a possible visualization of the rule in the Declare notation. Like for the case with three activities, the

problem remains that also subsets of the rules may hold for the process. For instance, rule generation also outputs the rules *{decide, examine casually}* $\rightarrow$ *{register request}* and *{examine casually}* $\rightarrow$ *{check ticket}*.

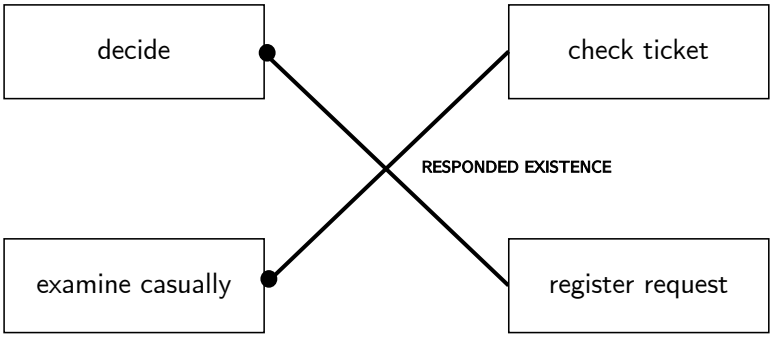


Figure 5.5: RESPONDED EXISTENCE Constraint for Four Activities (2 Premises and 2 Conclusions).

Theoretically, it is possible to have an infinite amount of items in the premises and conclusions. However, for reasons of comprehension and usefulness it most likely does not make sense to introduce rules or constraints with more than four activities in a single itemset. Besides, it is possible to create rules with one or three premises and one or three conclusions. Figure 5.6 and 5.7 visualize declarative constraints on the sample event log with one or three premises for a four items setting.

As introduced earlier, the association rule mining process can also include other optional attributes. Representing such association rules in the Declare modeling language requires the multi-perspective extension. Attributes are included in either the activation or correlation conditions depending on whether they appear in the antecedent, consequent, or both parts of the rule and how the process modeler decides to represent the association rule.

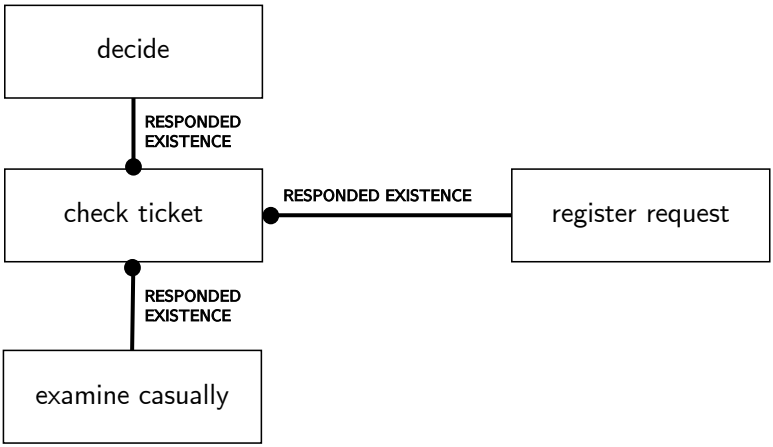


Figure 5.6: RESPONDED EXISTENCE Constraint for Four Activities (1 Premise and 3 Conclusions).

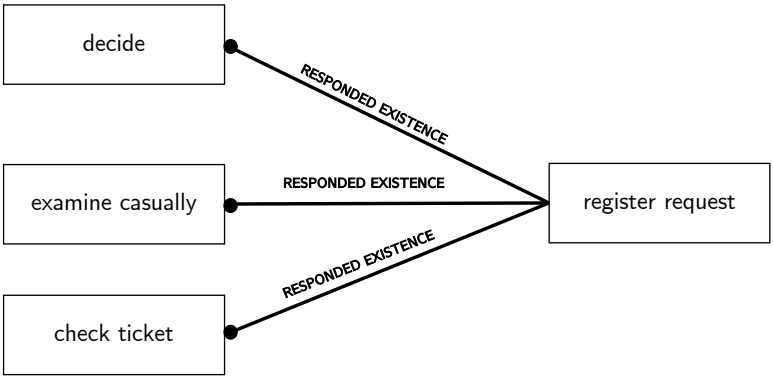


Figure 5.7: RESPONDED EXISTENCE Constraint for Four Activities (3 Premises and 1 Conclusion).

Figure 5.8 presents a multi-perspective declarative model representation of an association rule *{decide/Sara}* → *{check ticket}*. The constraint can be interpreted in a way that this **RESPONDED EXISTENCE** is only activated when resource *Sara* is deciding about the compensation request. There is no correlation condition (second brackets of the constraint are empty). Figure 5.9 shows a multi-perspective Declare translation for a rule of the form *{decide}* → *{check ticket/Sara}*. The only difference to Figure 5.8 is that the activation condition of the **RESPONDED EXISTENCE** constraint addresses the second activity of the constraint, indicated by *T.resource* instead of *A.resource*.

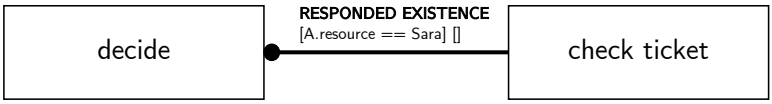


Figure 5.8: RESPONDED EXISTENCE Constraint with Activation Condition.

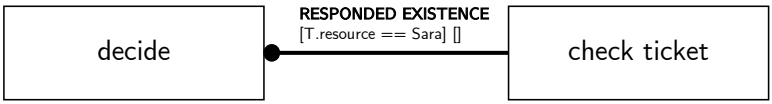


Figure 5.9: RESPONDED EXISTENCE Constraint with Activation Condition on Target Activity.

Another type of rule includes attributes on both sides of the constraint. Here, one should distinguish between cases where the attributes have different and equal values. The rules can be represented using either activation or correlation conditions in both cases. Consider an association rule *{check ticket/Mike}* → *{decide/Sara}*. Figure 5.10 shows one possible translation to the Declare notation using an AND-concatenation in the ac-

tivation condition of the **RESPONDED EXISTENCE** constraint. Another option is to introduce a correlation condition *different resource* for this constraint type. However, the disadvantage of this representation is that information about the concrete attribute values is lost. This means that the constraint generally enforces that two activities have to be executed by two different persons (resources), which may not be what is intended and what the association rule expresses.

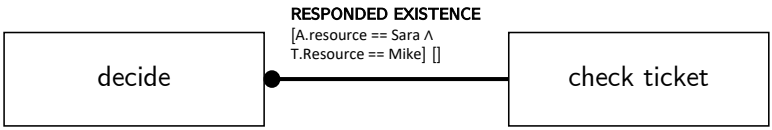


Figure 5.10: RESPONDED EXISTENCE Constraint with Two Activation Conditions.

Figure 5.11 depicts a Declare constraint for the rule *{check ticket/Sara}* $\rightarrow$ *{decide/Sara}*. It consists of a correlation condition *same resource* that indicates that both activities *decide* and *check ticket* have to be executed by the same resource. This general restriction applies to every resource associated with both activities.

Users without knowledge about the association rules do not exactly know from which single association rule or which association rules the Declare element originates. Alternatively, instead of using a correlation condition with the *same* indicator, association rules with the same attribute value on both sides of the rule can be translated to constraint with activation conditions like in Figure 5.10. The activation condition would be *A.resource = Sara* $\wedge$ *T.Resource= Sara*, whereby the whole expression has to evaluate to true to activate the **RESPONDED EXISTENCE** constraint.

It is crucial to consider their support when translating association rules including additional attributes to Declare constraints. The higher the

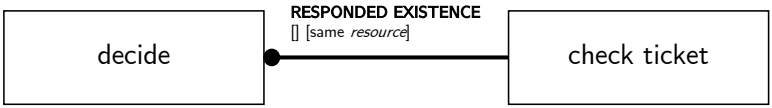


Figure 5.11: RESPONDED EXISTENCE Constraint with Correlation Condition.

support, the more often they are executed, and thus, related constraints represent a more general behavior of the process.

Some rule of thumb is the following: Rules with a low support (< 0.5) should never be translated to general constraints on the process, especially not to correlation conditions with *same* or *different* indicators. For association rules with a support from 0.5 to 0.8, modelers or process analysts should also tend to rather introduce activation conditions with concrete attribute values (e.g., A.resource = Mike $\wedge$ T.Resource= Sara instead of *different resource*). In exceptional cases and while considering domain knowledge for the process, persons may decide to transform it to a more general constraint when they know that other combinations of attribute values do not appear or are not relevant for the desired execution of a process. For association rules with attributes included and a support from 0.8 to less than one, there may be a rule to translate them to more abstract constraints generally.

However, if modelers or analysts have domain knowledge that prevents introducing a general statement and know about important cases which would not fulfill the constraint, they should also be transformed to more specific model elements in Declare. Finally, constraints with a support equal to one can always generate a universal constraint because it holds for every process instance in the event log. The support of an association rule is important to evaluate a constraint's relevancy and significance for the whole process and event log.

Modelers should try to introduce constraints based on rules with a support as high as possible. Besides the support, confidence values indicate the precision of a constraint regarding the full event log and a preferred direction of the association rules or Declare constraints. Consider rules of the form *{A}* → *{B}* and *{B}* → *{A}*, A and B are activities. Both rules have the same support because they consist of the same set of items. Assuming that the support of single item B is lower than the support of single item A, rule *{B}* → *{A}* has a higher confidence than *{B}* → *{A}* and should therefore be preferred for a transformation to a Declare model element. Because of the lower support of B, the resulting constraint is trivially (vacuously) satisfied in more cases than it would be with activity A in the antecedent part of the rule. This is preferable compared to incorrect descriptions of the process for more cases of the event log.

Besides **RESPONDED EXISTENCE**, association rules can also translate to **EXISTENCE** constraints. Thereby, no relationships between activities are formed but only a statement about the appearance of a particular activity is made. The reason for such a translation instead of introducing **RESPONDED EXISTENCE** constraints could be that, for instance, a relation between activities without any significance about their temporal order may not be meaningful or beneficial for a declarative model. Figure 5.12 shows the translation of the rule *{check ticket/Mike}* → *{decide/Sara}* to two Declare **EXISTENCE** constraints with activation conditions. For **EXISTENCE** constraints, association rules are not strictly required, but frequent itemsets are sufficient to derive these types of constraints. The support value alone can evaluate the suitability of an activity alone or in combination with attributes to be translated to an **EXISTENCE** constraint. As always, higher support values should be preferred.

Like for the **RESPONDED EXISTENCE** process restrictions, users should follow the general rule thumb that the higher a support value the more appropriate it is to represent it as a general condition for the whole

process. Figure 5.12 visualizes such constraint types where the *decide* and *check ticket* activities only have to be part of the process if the activation conditions hold. There is no information about the existence of these activities otherwise. The question arises whether such constraints represent a circular reference because whenever a value for the resource attribute of *decide* can be determined, it is clear that the activity exists. Though, one could argue to interpret them by using an alternative semantics: If, for instance, the *decide* activity exists, it should be associated with resource *Sara*.

Figure 5.12: Two EXISTENCE Constraints with Activation Conditions.

Association rules can only be translated to either **RESPONDED EXISTENCE**, **EXISTENCE**, or **CO-EXISTENCE** constraints and their multi-perspective extensions because no sequential order can be determined. This changes with sequential patterns. The following introduces the application of sequential pattern mining on event logs in RapidMiner and the translation of sequential patterns or rules to declarative model elements.

5.3 Sequential Pattern Mining on Event Logs

Sequential pattern mining on event logs allows considering and including the temporal order of activities in combination with additional attributes. Figure 5.13 presents a process of applying sequential pattern mining on event logs in RapidMiner [6]. The process uses the GSP operator

working as introduced in Chapter 4, also the event data input file remains unchanged. Like for association rule mining, the RapidMiner process consists of several preprocessing steps to bring the event data into a format suitable for the sequential pattern mining algorithm.

After importing the data, the date attribute of the event log has to be transformed from a date to a numerical attribute. The reason is that the sequential pattern mining operator in RapidMiner expects a numerical value for the time and does not support the use of attributes with type date. Each date value of the event log is transformed to one second relative to the epoch (start of the epoch is 1st of January 1970). Then, the *Activity*, *Case Id*, *Timestamp*, and *Resource* attributes of each entry of the event log are selected. Possible other attributes are discarded. Afterwards, the *Activity* and *Resource* attributes are transformed from a nominal to a binominal value. Clearly, it is possible to only transform the *Activity* and drop the *Resource* attribute.

Table 5.5 presents an excerpt of an event log after transforming the *Activity* and *Resource* attributes. For each different value of both attributes, a new column is inserted, and **true** or **false** values indicate whether the specific value is present for one instance of the process whereas only one activity and resource column is **true** for each line of the table. The more different values the event log has, the more columns have the resulting table. Triple points in Table 5.5 indicate that the representation of both activities and resources and the cases is not complete. Transforming the values is necessary because the GSP algorithm expects binominal attribute values.

Table 5.6 presents the parameters of the GSP operator in RapidMiner [6]. As introduced in Chapter 4, the GSP algorithm is frequently used for analyzing customers' markets baskets and their buying behavior. For an application in the process context, one instance of a process (case) is assumed to be one customer. Customers fill their shopping baskets and

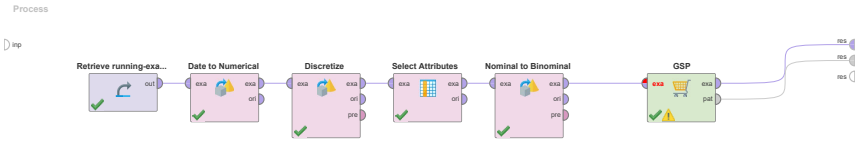


Figure 5.13: Process of Sequential Pattern Mining on Event Logs in RapidMiner [6].

buy certain products. In this use case, the activities of one process instance are treated as the set of products a customer buys. Therefore, the *customer id* is set to the *Case ID* attribute of the event log. The time attribute is set to the timestamp information of the event log in a numerical format relative to the epoch. In a first attempt, the minimal support for the sequential patterns is set to 0.8. This value can be varied and adapted depending on the user needs and expectations.

The GSP operator only produces sequential patterns that fulfill the minimum support. Next, the *window size* parameter defines a range of time in which activities are treated as a single transaction. For this process use case, the parameter is set to zero as there may be activities that happen seconds (or milliseconds) after another, and every activity should therefore be treated as single activity of the process.

There are two more parameters *max gap* and *min gap*, whereby the former determines a maximal difference in time in which the GSP algorithm generates sequential patterns. Patterns with activities whose time difference is too big will not be included in the same pattern. Similarly, the *min gap* operator defines a minimum time difference of the activities. For the first run, the *maximum gap* is set to Infinity. In this way, the algorithm generates patterns for all activities, no matter how big the time difference between them is. The minimal gap is set to zero because it is desirable

Table 5.5: Event Log Excerpt in Binominal Format (Activity and Resource Attributes).

register request	check ticket	...	Pete	Sue	...	Case ID	Time
true	false	...	true	false	...	1	1293703320
false	true	...	false	true	...	1	1293786360
...	...	...	...	...	...	...	...
true	false	...	false	false	...	2	1293705120
false	true	...	false	false	...	2	1293707520
...	...	...	...	...	...	...	...
true	false	...	true	false	...	3	1293715920
false	false	...	false	false	...	3	1293723240
...	...	...	...	...	...	...	...
true	false	...	true	false	...	4	1294322520
false	false	...	false	false	...	4	1294398360
...	...	...	...	...	...	...	...
true	false	...	false	false	...	5	1294300920
false	false	...	false	false	...	5	1294391760
...	...	...	...	...	...	...	...
true	false	...	false	false	...	6	1294322520
false	false	...	false	false	...	6	1294326360
...	...	...	...	...	...	...	...

also to find patterns for activities that happen within a time window of one millisecond.

Finally, the *positive value* parameter is set to *true*. This parameter value indicates the binominal value that shall be treated as positive. Here, the Boolean value *true* was used. However, other encodings like zero or one are conceivable. With these GSP operator settings and the complete picture shown in Figure 5.13, the process is ready-to-run.

Table 5.6: GSP Operator Parameters in RapidMiner [6].

Parameter	Value
customer id	Case ID
time attribute	dd/MM/yyyy:HH.mm
min support	0.8
window size	0.0
max gap	Infinity
min gap	0.0
positive value	true

Table 5.7 shows a subset of the output for sequential pattern mining using the GSP algorithm on the compensation request process event data. The process generates sequential patterns with a minimum support of 0.83 (0.8 was specified) and up to three transactions and four items and finds 20 patterns in total. Thereby, a transaction is one element of the sequential pattern. One transaction can be composed out of two elements (possibly more for other scenarios and datasets), whereby one of them is an activity and the other one an information about used resources. Patterns, where an activity/resource combination is followed by a single activity or the other way around, are also possible. This means that an activity follows or precedes an activity with the respective resource information. Furthermore, there can be sequential patterns where an activity follows or precedes a resource attribute alone or even resource attributes following each other.

Contrary to the association rule mining process, with this sequential pattern mining approach, it is impossible to directly specify the maximum or minimum number of items a sequential pattern can consist of. Instead, the shape of the output results from the GSP operator parameter

settings and the number of binominal attributes and can only be indirectly influenced. One can derive sequential rules from such sequential patterns to benefit from confidence measures and an alternative format for practitioners.

Table 5.7: Excerpt of the GSP Output for the Compensation Request Process Dataset.

Support	Transactions	Items	T1	T2	T3
0.833	2	2	register request	Ellen	
0.833	2	2	Mike	Ellen	
1	2	2	register request	check ticket	
1	2	2	register request	decide	
1	2	2	check ticket	decide	
1	2	2	decide/Sara		
1	2	2	Mike	decide	
1	3	3	register request	check ticket	decide
1	3	3	register request	check ticket	Sara
1	2	3	check ticket	decide/Sara	
1	3	3	register request	check ticket	decide/Sara
...	...	...	...	...	....

5.3.1 Translating Sequential Patterns to Sequential Rules

There is only one possibility to form a sequential rule for sequential patterns with only two transactions. For instance, the pattern *{register request, check ticket}* is translated to a rule *{register request}* $\rightarrow$ *{check ticket}* and has a confidence of $\frac{1}{1} = 1$ as activity *register request* appears in a hundred percent of the cases. Similarly, the pattern *{Mike, Ellen}* forms a sequential rule *{Mike}* $\rightarrow$ *{Ellen}* with a confidence of $\frac{0.833}{1} = 0.833$. Only in one case of the process, resource *Mike* is associated with the execution

of some activity of the process while not being followed by an activity associated with resource *Ellen*.

Things are only slightly more complex for patterns with more than two transactions. The number of items as depicted in Table 5.7 does not influence the rule generation process. A transaction itself, possibly a combination of an activity and additional attributes like resource information, is kept together and not split for the transformation to a sequential rule (e.g., *{check ticket}* $\rightarrow$ *{decide/Sara}*). The Pattern *{register request, check ticket, decide}* can be transformed to either *{register request, check ticket}* $\rightarrow$ *{decide}* or *{register request}* $\rightarrow$ *{check ticket, decide}*. Both rules have the same confidence (equal to one) because *{register request, check ticket}* as well as *{register request}* alone appear in all instances of the process. If confidence values are different, the rule with the higher value should generally be preferred. There are even more possible rule transformations for sequential patterns with four transactions. They can be represented by a rule with three transactions in the antecedent and one in the consequent part, one transaction in the antecedent and three in the consequent part, or two transactions in both.

As usual, higher confidence values are preferable. However, there can also be cases where modelers may decide to consider rules with lower confidence values because he or she believes that they are more important based on domain knowledge about the process and overall setting. Theoretically, there can be patterns with more than four activities; however, the question arises whether such large structures are still meaningful and can be interpreted by users working with a particular process.

In the last step, the sequential patterns or rules must be translated to Declare constraints to include them into a declarative process model and finish the process mining procedure. The following introduces a selection of the most common transformation approaches for sequential patterns and rules, similar to the ones for association rule mining.

5.3.2 Translating Sequential Patterns or Rules to Declare Constraints

In the simplest case, the sequential pattern mining algorithm outputs a pattern with two transactions and two items, whereby both items are activities. This pattern can be translated to a **RESPONSE** constraint in the Declare notation, meaning that whenever the first activity is executed, the second one has to follow eventually. Figure 5.14 shows a graphical representation of the **RESPONSE** constraint in Declare based on the *{register request, check ticket}* pattern in Table 5.7.

Figure 5.14: RESPONSE Constraint for Sequential Pattern with Two Activities.

A similar transformation approach is possible for patterns with three transactions and three items, all of them being an activity. Figure 5.15 shows the graphical representation in the Declare notation with three activities, based on the same data as before. The former example represents a subset of the transformation shown here.

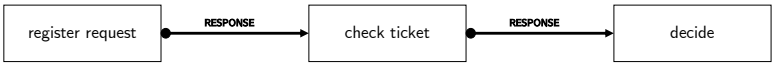


Figure 5.15: Chained RESPONSE Constraints for Sequential Pattern with Three Activities.

Theoretically, it is possible to have a pattern with an infinite amount of activities, each of them representing a single transaction, and then transform them to a chain of **RESPONSE** constraints. However, the

more activities a pattern consists of, the more confusing it potentially is. Thereby, the advantages and strengths of the declarative representation compared to an imperative model in, for instance, Petri net or BPMN notation would not emerge sufficiently. Figure 5.16 shows a declarative representation of the same pattern as in Figure 5.15 assuming that it is interpreted as a rule *{register request}* $\rightarrow$ *{check ticket, decide}* by extending the original Declare notation set. The additional challenge here, compared with transforming association rules, is to avoid the loss of information about the sequential order of activities.

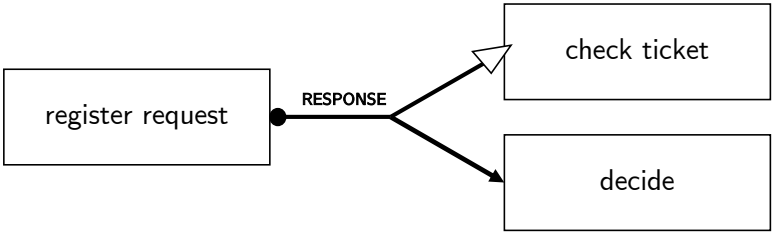


Figure 5.16: Unchained RESPONSE Constraint for Sequential Pattern with Three Activities (1 Premise, 2 Conclusions).

Figure 5.16 demands that whenever activity *register request* appears, both activities *check ticket* and *decide* have to be executed eventually afterwards. The blank arrow leading to the *check ticket* activity indicates that it still always precedes *decide* even though both are part of the consequent of the sequential rule representation. Figure 5.17 depicts the opposite case where the appearance of two activities leads to execution of another activity at a later time. The blank circle at the *register request* activity indicates that it precedes *check ticket*. In this way, the information about the overall sequential order can be preserved.

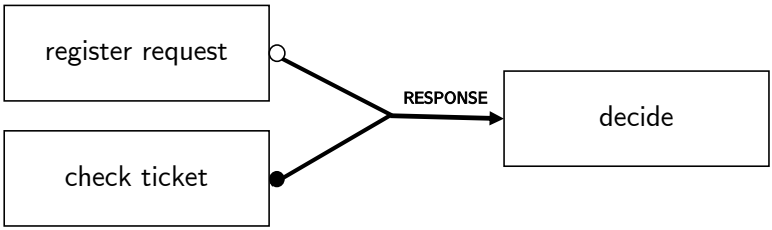


Figure 5.17: Unchained RESPONSE Constraint for Sequential Pattern with Three Activities (2 Premises, 1 Conclusion).

Finally, the model elements in Figure 5.16 and 5.17 can be combined into sequential rules with two activities in both antecedent and consequent. Figure 5.18 shows such a constraint in adapted Declare notation. It results from the sequential pattern *{register request, check ticket, decide, reject request}* (not included in Figure Table 5.7), transformed to the sequential rule *{register request, check ticket}* $\rightarrow$ *{decide, reject request}* and has a support of 0.5.

Modelers will most likely not include this pattern in their Declare model because in 50 percent of the cases, the course of the process follows the pattern *{register request, check ticket, decide, pay compensation}*, indicating rejected compensation requests.

Constructs with more than four activities are theoretically possible. However, the question arises whether they are still useful for being included in a declarative model. Such large sequences should most likely be split up into, for instance, constructs of three and two activities. Again, domain knowledge about the process can help find the most appropriate representation that delivers the most benefit for users to understand the process while keeping the model lucid.

Consider that still combining **RESPONSE** constraints does not have the semantics as a description in a declarative process modeling notation like

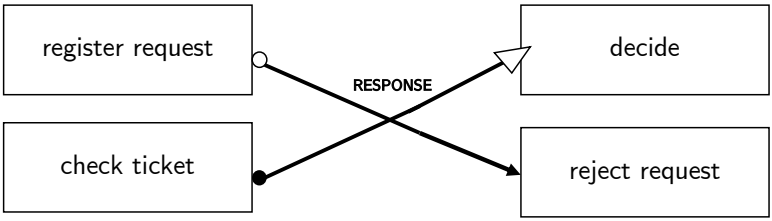


Figure 5.18: Unchained RESPONSE Constraint for Sequential Pattern with Four Activities (2 Premises, 2 Conclusions).

BPMN or Petri nets. The **RESPONSE** constraints, in contrast to control flow descriptions, do not enforce the execution of two activities directly after each other, though, **CHAIN RESPONSE** or **CHAIN SUCCESSION** constraints can do this. However, such semantics are not intended here and cannot originally be derived from the GSP output.

The previous transformations and model constructs are all based on sequential patterns that only consist of activities, with no additional attributes included. **RESPONSE** constraints are adequate to depict the sequential order of the process activities. However, patterns where such constraints are included, may often be more interesting for users. The multi-perspective Declare extension is very capable of representing them. Therefore, the following provides sample translations based on the types of patterns shown in Table 5.7.

The most simplistic pattern with an activity and additional attribute proportion consists of only one transaction but two items whereby one item is an activity and the other an optional attribute such as resource information. Line 6 of Table 5.7 holds such a pattern. This constraint type requires the multi-perspective extension of Declare (MP-Declare) like used in the *RuM* tool [ADCH$^+$]. Optional attribute information cannot be translated to activation or correlation conditions of multi-perspective

Declare constraints here. The sequential pattern *{decide/Sara}* can be translated to an **EXISTENCE** constraint for the *decide* activity including an activation condition. It may, similar to the multi-perspective **EXISTENCE** case in association rule mining, be interpreted as a demand that every appearance of the *decide* activity should be associated to resource (i.e., person or employee) *Sara*.

Figure 5.19 shows such a constraint in Declare notation. Here, introducing the **EXISTENCE** constraint without any activation condition would also be uncritical since the pattern has a support of one. For cases of lower support, like for the **RESPONDED EXISTENCE** and **RESPONSE** constraint types, users should carefully consider whether they want to include it in the declarative model because no activation condition prevents the general validity statement that comes with it.

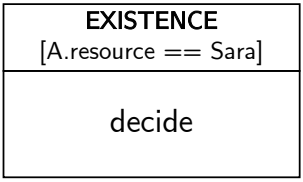


Figure 5.19: EXISTENCE Constraint for *decide* Activity with Activation Condition.

Another type of sequential pattern consists of an activity/attribute combination followed or preceded by a single activity. One representative of such a pattern is *{check ticket, decide/Sara}*, shown in line 10 of Table 5.7. In multi-perspective Declare models, such patterns are represented by a **RESPONSE** constraint with an activation condition between the *check ticket* and *decide* activity. Figure 5.20 depicts such a construct. Variants of this model element are analogue to the multi-perspective **RESPONDED EXISTENCE** representations for association rules (Figure 5.8 to 5.11).

If the resource information is part of the first activity, the activation condition of the **RESPONSE** constraint is *A.resource == Sara*.

Figure 5.21 shows a Declare **RESPONSE** constraint with resource information for both activities. It originates from the sequential pattern *{register request/Pete, decide/Sara}*. Since it has a support of only 0.33, it is somewhat unlikely to be included in the declarative model. Nevertheless, it nicely illustrates the case where a sequential pattern consists of two activity/attribute combinations. Like for the **RESPONDED EXISTENCE** constraint based on association rules, there are two possibilities to represent them using either activation or correlation condition. Figure 5.21 depicts the activation condition case while the correlation conditions with *same* or *different* identifiers is used analogue to Figure 5.11, just replacing the **RESPONDED EXISTENCE** with a **RESPONSE** constraint.

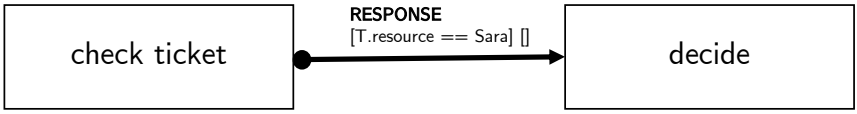


Figure 5.20: RESPONSE Constraint for Sequential Pattern with One Activity/Attribute Combination.

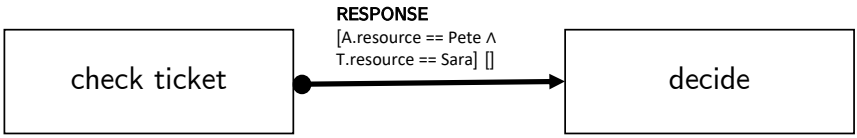


Figure 5.21: RESPONSE Constraint for Sequential Pattern with Two Activity/Attribute Combinations.

Multi-perspective Declare model elements can also represent sequential patterns with more than two activities. Consider the pattern *{register*

request, check ticket, decide/Sara} with a support of one. A graphical visualization results in constraint like depicted in Figure 5.22.

The **RESPONSE** constraint between the *check ticket* and *decide* activities is only active in case resource *Sara* is associated with the *decide* activity. Moreover, it is independent of the **RESPONSE** constraint between *register request* and *check ticket*. This means that the Declare process model (or the specific model part in Figure 5.22) could (in case *Sara* is not associated with *decide*) enforce a **RESPONSE** relation between *register request* and *check ticket* but not between *check ticket* and *decide*. Like for the non-multi-perspective case without the additional attributes, the patterns can be transformed to sequential rules that are finally represented as unchained model constructs like in Figures 5.16 to 5.18.

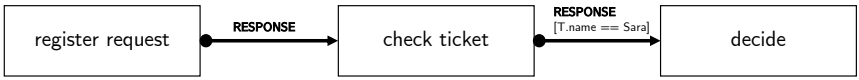


Figure 5.22: Chained RESPONSE Constraints with Multi-perspective Part.

Things are only slightly more complex for the multi-perspective case here because the activation and correlation conditions of a binary constraint can address more than the activation (abbreviated with A) and target (abbreviated with T) activities (there are three activities). Activities on the left and right sides of the rule have to be differentiated from each other. One pragmatic approach is to number them in the form of A_1, A_2, T_1, and T_2. Figure 5.23 presents an alternative representation of Figure 5.22. The sequential pattern *{register request, check ticket, decide/Sara}* is interpreted as a sequential rule *{register request, check ticket}* $\rightarrow$ *{decide/Sara}* with a support and confidence of one.

Like before, the blank and filled dots indicate that *register request* precedes *check ticket* in the sequential order. Unlike before, the whole con-

struct is only active when the activation condition *T.resource == Sara* is fulfilled, nothing is said about the relation between *register request* and *check ticket* in case the condition does not hold. Again, there is a tradeoff between the preciseness and detail degree and the loss of information for advanced model elements.

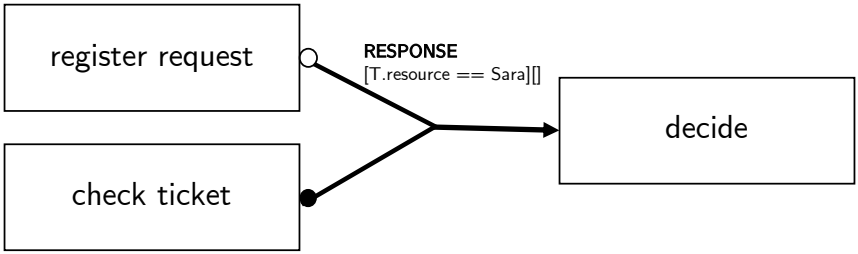


Figure 5.23: Unchained RESPONSE Constraints for Three Activities (Multi-perspective).

Like for the standard Declare model constructs, similar to Figures 5.16 and 5.18, also rules with only one activity in the antecedent and two in the consequent as well as two in both parts can be represented. Figure 5.24 shows a Declare **RESPONSE** construct for four activities, two in both parts of the underlying rule, in a generalized way. Even when dramatically decreasing the minimum support of the GSP operator, the RapidMiner process does not find sequential patterns with four activity/attribute combinations. Still, this could happen for other (most likely larger) processes. Therefore, Figure 5.18 provides an appropriate pattern for their translation to a Declare construct. As before, blank dots and arrows indicate the precedence relationship between the activities in the antecedent and consequent parts. With four activities involved, the activation condition can consist of an *AND*-concatenation of up to four expressions if the sequential pattern contains four activity/attribute combinations. The *attr*

part in the expression refers to the name of the attribute, for instance, *resource* like in the examples shown previously. Correlation conditions can also be used to describe the relationship between the attributes associated with the activities, for instance, using *same* or *different* identifiers. A condition *same resource* affects four activities at once. Therefore, correlation conditions of, for instance, the form $T_1.resource == T_2.resource$ or $T_3.resource != T_4.resource$ are probably more likely to appear in practice.

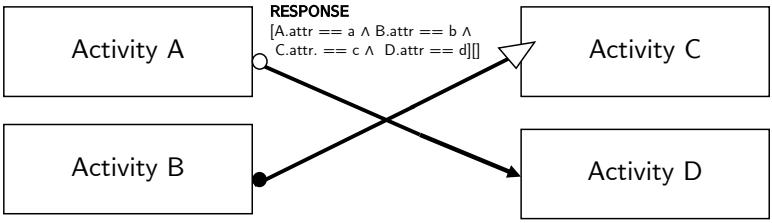


Figure 5.24: Generalized Unchained RESPONSE Constraint (Four Activities).

Another special type of sequential patterns resulting from the GSP process includes optional attributes standing alone, without being associated with an activity. Such attributes can be followed or preceded by any combination of other elements. These patterns occur because activities and other optional attributes are both equally input to the GSP algorithm. An example is *{Mike, decide}* which can be derived from line 7 of Table 5.7. Contentwise, it describes that the appearance of resource *Mike* at any time in the process must eventually be followed by the *decide* activity. The question arises whether such constraints are meaningful and benefit the understanding of the underlying process.

There may, however, be processes where such a relationship is important (e.g., special cases of the process that definitely have to be detected to treat them accordingly). Figure 5.25 shows how such a pattern can

be represented by extending the MP-Declare notation set using a standard **RESPONSE** constraint with an activation condition demanding that resource *Mike* is associated with the activation activity with the difference that the activating activity is not concretely specified. Any activity with resource *Mike* associated can activate this constraint. The reverse constraint with the blank activity on the right side of the **RESPONSE** is also possible. Moreover, applying the GSP operator on the compensation request process event log results in patterns that only consist of resources such as *{Mike, Sara}*. Such patterns are not meaningful in the process context and should normally not be considered further for a declarative representation.

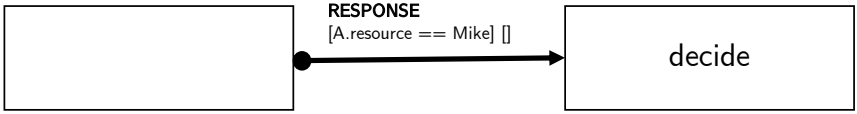


Figure 5.25: RESPONSE Constraint for Sequential Pattern with Standalone Resource Attribute.

All previously introduced sequential rules and patterns with attributes (as the one represented in Figure 5.25) contain attributes that are solely related to one particular activity, so-called event attributes. In practice, however, case attributes related to one execution of a process (i.e., case) may also play an important role. Even though the RapidMiner process as depicted in Figure 5.13 does not directly support case attributes, there can be de-facto case attributes whenever, on the one hand, the attributes themselves and on the other hand, their values are equal for every activity belonging to one instance of the process. Furthermore, process modelers may want to enrich the declarative model with their observations and knowledge about the domain. Thus, there should be a possibility to represent relationships including case attributes.

Figure 5.26 shows one proposal of how to include case attributes in Declare model constructs. Case attributes are visualized with hexagons whereby their names and values are placed inside the hexagons. The example shown here is artificially constructed and cannot be derived from the original compensation request process event log or the output of the GSP RapidMiner process. Consider an event log that contains another attribute *Risk* with possible values *Low*, *Medium*, and *High* whose value is based on historical customer and insurance case data assessing the risk or likeliness that some compensation request may be a fraud. Then, the GSP operator may generate a sequential pattern *{High, examine thoroughly}* meaning whenever the risk was assessed to be high, there was a more detailed inspection of the case in the past.

There is no temporal order between case attributes and activities; thus, the construct adopts the **RESPONDED EXISTENCE** connection type. Another type of sequential pattern with case attributes is a case followed by two activities. An example for such a pattern is *{High, examine thoroughly, decide}*, interpreted as a rule *{High}* $\rightarrow$ *{examine thoroughly, decide}*. In this case, there is an **EXISTENCE** relation between attribute *Risk* with value *High* but also the **RESPONSE** relation between *examine thoroughly* and *decide* remains and should be depicted in the Declare model. A user should never forget that there cannot be any relations between case attributes and activities that involve temporal orders to avoid misinterpretation.

In the third type of pattern, the case attributes are combined with activities. For instance, pattern *{examine thoroughly/High, decide}* could be transformed to the Declare construct shown in Figure 5.28. To differentiate between event and case attributes, the *Risk* attribute part of the pattern's first item does not lead to an activation condition of the **RESPONSE** constraint but is depicted as an independent hexagon attached to the constraint's connection line. If the case attribute *Risk* with value *High* was combined with the decide activity, the dot at the line connecting the

case attribute with the **RESPONSE** constraint would be replaced by an arrow pointing towards the hexagon. In this way, it is made clear that the *Risk* attribute is combined with the *decide* activity. Patterns with single case attributes at the last place or in the consequent part of a sequential rule are not meaningful and should not be considered for being included in a declarative process model.

Furthermore, like for event attributes, process modelers should neglect patterns that only consist of case attributes without any attachment to activities. They most likely do not deliver any useful information about the process. All meaningful sequential patterns and rules that include case attributes can be translated to appropriate Declare constructs. Even though the RapidMiner process does not explicitly support them, they can deliver valuable information when integrated into a declarative model.

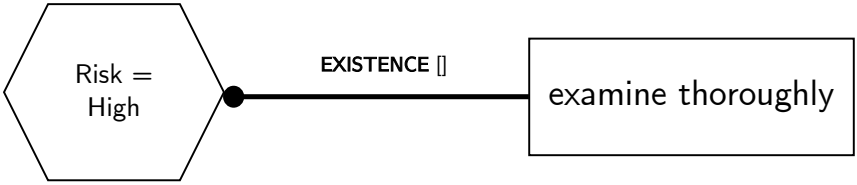


Figure 5.26: EXISTENCE Constraint for Sequential Pattern with Case Attribute.

Some tools like the MP-Declare Editor of the RuM application [ADCH$^+$] additionally allow introducing time conditions between the activities. A time condition describes the time range in which the second activity is executed after the first, e.g., [2,5,h] means that an activity has to be executed between two and five hours after the first one. However, it is impossible to generate such conditions from an event log with the RapidMiner process shown. The reason is that the timestamp information cannot be translated to a binominal value. It is used as the time attribute

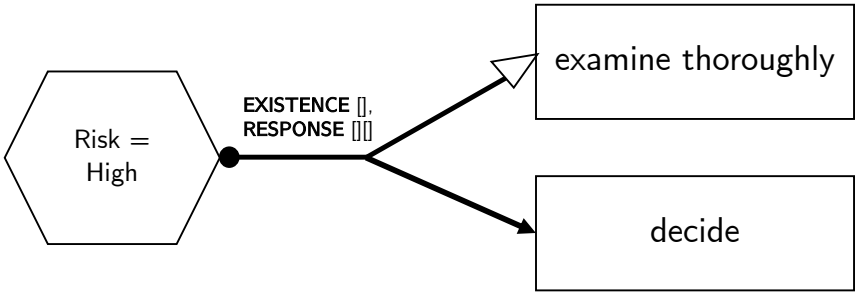


Figure 5.27: EXISTENCE/RESPONSE Constraints for Sequential Pattern with Case Attribute and Two Activities.

of the GSP algorithm but not as a defining attribute that contributes to understanding the (data-related) relationships of a process.

Moreover, with the output of the sequential pattern at hand, it is not possible to introduce constraints with negative semantics, such as **ABSENCE** or **NOT RESPONSE**. The simple reason is that association rule and sequential pattern mining algorithms can only detect existing patterns and rules and none that are not. This is a general shortcoming of process mining: it cannot generate analyses of absent behavior, which, however, is highly desirable to exploit the advantages of the declarative modeling paradigm fully. Introducing this information into a process model often requires human knowledge and interaction or more advanced algorithms.

Also, the sequential patterns or rules cannot lead to introducing some of the strengthened constraints in the Declare notation set, such as **ALTERNATE RESPONSE**, **ALTERNATE PRECEDENCE** or **CHAIN SUCCESSION**. The reason is that **ALTERNATE RESPONSE** and **ALTERNATE PRECEDENCE** demand that once some activity appears, another activity must directly follow or precede without any other activities allowed in between. This setting, however, cannot be derived from the GSP output

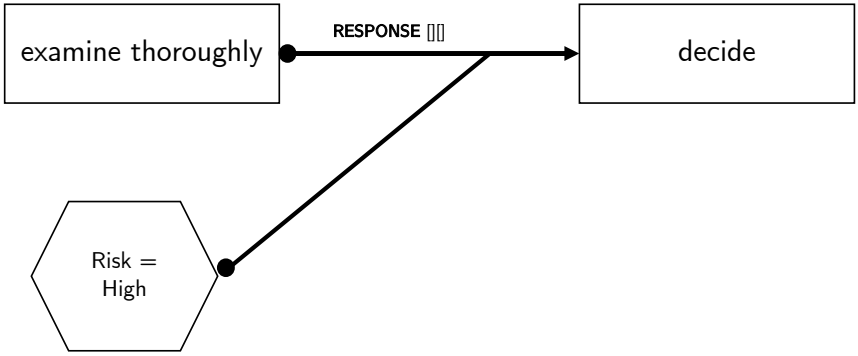


Figure 5.28: RESPONSE Constraints for Sequential Pattern with Case Attribute/Activity Combination.

because the sequential patterns can skip activities that do not appear in the patterns but are still part of the process. The **CHAIN SUCCESSION** constraint is a logical conclusion when both other constraints hold; therefore, it is also not possible to introduce it based on the output of the GSP algorithm.

All the previously introduced transformation approaches of sequential rules or patterns to Declare model elements with all kinds of variants have in common that they result in single instances or combinations of **EXISTENCE**, **RESPONDED EXISTENCE** or **RESPONSE** constraints. However, there is also **PRECEDENCE** as another main type of constraint in the Declare notation set. Consider a sequential rule with two activities (without any optional attributes) like *{register request, check ticket}*. In contrast to **RESPONDED EXISTENCE** or **RESPONSE**, the **PRECEDENCE** constraint is activated by the activity that appears later, meaning that it is part of the consequent in a sequential rule. When the single activating activity or the multiple activating activities are not part of a

process execution, the **RESPONSE** constraint is not active, meaning that it does not introduce any restrictions on the process.

One can estimate the preciseness of the constraint with the confidence of the underlying sequential rule. For **PRECEDENCE**, the support of the underlying sequential pattern is known, but one cannot estimate in how many cases the second activity is present without the appearance of the first activity and in how many cases the second is also not present at all. Therefore, it is not possible to assess the preciseness of a **PRECEDENCE** constraint, and process modelers should be cautious about including them into a declarative model, especially when the support of the underlying pattern is not that high.

There are two exceptions. If the support of the underlying sequential pattern is one, it is immediately apparent that both activities appear together in all instances of the process, and it behoves the modeler's design choice whether he or she wants to include a **RESPONSE** or **PRECEDENCE** constraint. Moreover, if the confidence of the reverse sequential rule (in this example *{check ticket}* $\rightarrow$ *{register request}* is known and high (preferably close to or exactly one), introducing a **PRECEDENCE** constraint is also uncritical. While the confidence of the reverse rule may be easy to determine for only two activities, things will get more complicated for larger rules with possibly several activity/attribute combinations. For these reasons, however, the use of **RESPONSE** should be the first choice and will be more frequent. Figure 5.29 shows a graphical representation of a **PRECEDENCE** constraint in Declare notation for the *register request* and *check ticket* activities.

All previous sample patterns, rules, and transformations use the *resource* attribute as it is a typical example for an event attribute and frequently found in process event logs. The RapidMiner process with the GSP algorithm works the same for other nominal event attributes. Things are only slightly more complicated for non-nominal types of attributes. One very

Figure 5.29: PRECEDENCE Constraint for Sequential Pattern with Two Activities.

common information in an event log describing process executions is an activity execution's costs. Costs are typically stated in a numerical value in a specific currency (e.g., dollars, euros).

In such a format, cost attributes are not suitable for analysis with sequential pattern mining because there are too many (possibly infinitely) different cost values that would result in an infinite amount of binominal values. Therefore, the values of the cost attribute have to be discretized before using them in the GSP algorithm. A typical approach for discretization is to introduce ranges of costs that evaluate their level in the process context. For the compensation request process, in a first approach, the following ranges are selected:

- Very low costs: 0-50
- Low costs: 51-100
- Medium costs: 101-200
- High costs: >200

In the same way as the resource attribute, the categories of costs have to be transformed to a binominal attribute. When adding using both the *costs* and *resource* attribute at the same, the GSP operator produces a great number of patterns. The *costs* attribute can also be used alone. Sorting the list so that patterns with more items are at the top lets the users instantly

see patterns with up to four activity/attribute combinations and a support of one. Besides, the RapidMiner process outputs the following transaction pattern with a support of one:

{register request}, {check ticket/low}, {decide/Sara/medium}, {medium}

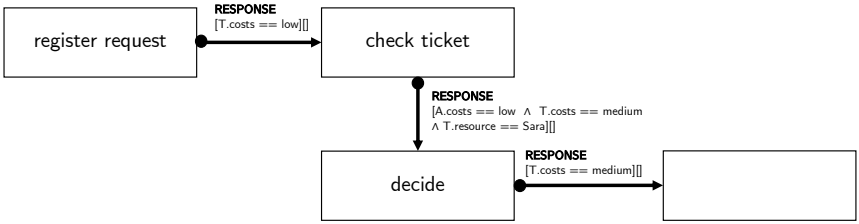


Figure 5.30: Chained RESPONSE Constraints with Costs Attributes.

Figure 5.30 shows a straightforward translation of this pattern to a multi-perspective Declare model. It demonstrates that the multi-perspective Declare elements work the same for one, two, or even more attributes associated with an activity. The more attributes are part of a combination, the more complex and potentially confusing the model. The use of correlation conditions is possible and advisable here, for instance using the *different* identifier for the costs of *check ticket* and *decide* or *same* for *decide* and the unknown activity. Alternatively, the four-activity construct depicted in Figure 5.18 is eligible for representing this pattern or rule.

Chapter 6 will directly take up the transformations shown in this chapter by applying them all together. It shows a full declarative model for the compensation request and other, larger, and more complex processes like production or O2C that are more similar to real-world processes in the industry. Simultaneously, the chapter will show how process modelers and various other stakeholders of the process could benefit from the

declarative representation. Also, it provides recommendations for operator settings and attributes inclusion in RapidMiner and more detailed design and transformation choices for the resulting association rules and sequential patterns.

6 Evaluation

This chapter evaluates the approach presented in Chapter 5. In the first step, the evaluation comprises an application of association rule and sequential pattern mining on three sample event datasets. They vary in the domain of the process they describe and a size ranging from the already known insurance compensation request process to a large real-world purchase order process of a multi-national company with more than one million events. For each application, the necessary steps, including pre-processing of the operators' event data and settings, are shown. Then, the resulting patterns and rules are translated into a Declare notation representation. This can result in independent constraints or a complete, coherent declarative process model depending on whether connections between the most relevant rules and patterns can be established.

Finally, the constraints and patterns are evaluated regarding their usefulness in the domain context and their understandability for various stakeholders. In this context, stakeholders are all involved in a particular business process, e.g., managers, process owners, process analysts, and employees performing human labor for one more activity of this process. The ultimate goal is to produce an evaluation of whether and how the data mining techniques can improve process discovery using the declarative modeling paradigm. There is a general discussion of the results in the end.

6.1 Evaluation Method

The evaluation follows a pragmatic approach meaning that it provides best-effort answers to questions addressing the application of association rule and sequential pattern mining to event logs. There is no real-world study with many companies; however, a detailed look at the RapidMiner output and the transformations to Declare constraints can already provide an adequate impression of the potential benefits. One aspect of the evaluation determines whether there is the general possibility of applying the techniques and, secondly, whether they produce an output that provides the ability to derive relationships about the process. Thereby, the evaluation checks on the one hand whether the applications are technically possible whereby performance issues caused by too large datasets could theoretically constitute an obstacle here.

One the other hand, the chapter investigates whether the RapidMiner processes find a sufficient number of rules and patterns with reasonable support and confidence values. A second aspect measures the usefulness of these rules and patterns to understand the processes themselves or their suitableness for constructing process-related Declare constraints. Here, the goal is to evaluate whether the rules and patterns can collectively form a meaningful Declare representation that includes coherent model elements or they do not have any connection to each other. The following shows the application of the approach in this thesis on three event logs from web repositories.

6.2 Application on Sample Event Data

First, a very small compensation request process serves as a blueprint for a straightforward application of the approach. Second, a more extensive dataset describing the execution of a larger production process is used

as an example for applying the fundamental data mining techniques on data exports that are similar to datasets in practice. Third, the approach is applied to a multi-national production firm's very large real-world purchase

6.2.1 Compensation Request Process

The first sample dataset is very small and publicly available on the web [9]. It consists of process instances from a synthetic insurance claim or compensation request process. Table 6.1 shows an overview of its properties. Both *Resource* and *Costs* are event attributes, i.e., they describe which (human) resource was responsible for executing a step and how much costs that caused. This particular dataset is used to demonstrate the approach's applicability generally; the benefit of understanding the domain and process through the approach in this thesis compared to standard process mining applications is expected to be relatively low.

Table 6.1: Key Facts of Compensation Request Process.

Domain	Insurance claim process
#Activities	8
#Cases	6
#Activity Executions	42
Optional Attributes	2 (Resource, Costs)

The log describes the execution of a compensation request or insurance claim process with eight activities. Figure 3.10 and 3.11 in Chapter 3 show process models discovered from the event data using the Disco tool [4]. In the happy path case, after registering, a request is either examined casually or thoroughly. Then, the ticket is checked and forwarded to the

decision step, and the compensation is paid. Alternatively, the deciding person initiates a re-initiation of the request, e.g., when there is a need for another examination or check of the ticket. Furthermore, the insurance company can reject requests.

Association Rule Mining

Table 5.2 and 5.1 in Chapter 5 present intermediate results of the frequent itemsets generation and the association rules that result from them. The minimum number of frequent itemsets is set to ten. When not enforcing any minimum number of frequent itemsets and setting the minimum support to 0.6, the minimum and maximum numbers of items per itemset to two and three, the operator finds the four frequent patterns shown in Table 6.2. Based on these itemsets, twelve association rules (shown again in Table 6.3) are constructed using a minimum confidence setting of 0.6.

Table 6.2: Frequent Itemsets for Compensation Request Process (No Minimum).

Size	Support	Item 1	Item 2	Item 3
2	1.000	check ticket	register request	
2	1.000	check ticket	examine casually	
2	1.000	register request	examine casually	
3	0.667	check ticket	register request	examine casually

All association rules have a support and confidence value of one. As stated in Chapter 5, association rules cannot determine anything about the temporal order of the activities. They can be translated to **EXISTENCE**, **CO-EXISTENCE**, and **RESPONDED EXISTENCE** constraints (**PRECEDENCE** in exceptional cases) in the Declare notation. Figure 6.1 presents one possible translation of the association rules to a declarative process model. With these settings of the RapidMiner operators, the set

Table 6.3: Association Rules for Compensation Request Process.

Premises	Conclusion	Supp.	Conf.
check ticket, decide	register request	1.0	1.0
check ticket, register request	decide	1.0	1.0
decide, register request	check ticket	1.0	1.0
check ticket	decide	1.0	1.0
decide	check ticket	1.0	1.0
check ticket	register request	1.0	1.0
register request	check ticket	1.0	1.0
decide	register request	1.0	1.0
register request	decide	1.0	1.0
check ticket	decide, register request	1.0	1.0
decide	check ticket, register request	1.0	1.0
register request	check ticket, decide	1.0	1.0

of association rules only contains three activities (*register request*, *check ticket*, *decide*) with each of them being in the antecedent or consequent of the rule in many variants. One transformation between the association rules and the Declare notation can introduce an **EXISTENCE** constraint for each activity. This can be derived from a single rule that contains the three activities, e.g., *{check ticket, decide}* $\rightarrow$ *{register request}*, with a support value of one. Each of the three activities has a **RESPONDED EXISTENCE** relation with the two other activities in both directions. The two lines representing the **RESPONDED EXISTENCE** relation show that both directions of the constraint hold; for visualization reasons they could be combined into one connection with dots at both ends.

For every combination of the three activities, there is an association rule with one activity as the premise and the others as the conclusion that has

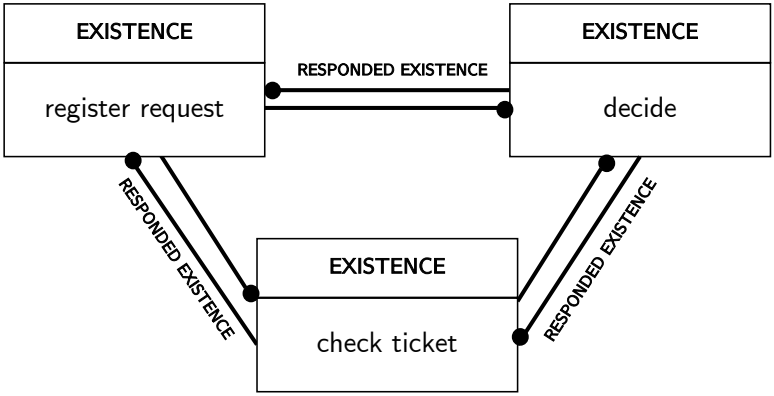


Figure 6.1: EXISTENCE and RESPONDED EXISTENCE Constraints for Compensation Request Process.

a support and confidence value of one. Thus, they could theoretically be represented by a construct like shown in Figure 5.3 whereby, it makes most likely the most sense to use *{register request, check ticket}* $\rightarrow$ *{decide}* as the underlying rule. The model in Figure 6.1, however, has the advantage that it depicts all the relations among each of the three activities. Even though the temporal order of the three activities is apparent, it cannot originally be derived from the association rules. Overall, support values alone are the decisive factor here, leading to confidence values of one as well. There are no cases in which one of the activities is not present; thus, the **RESPONDED EXISTENCE** constraint is perfectly appropriate.

The declarative model shown in Figure 6.1 can be reduced to a model that only consists of **CO-EXISTENCE** constraints. Figure 6.2 presents such an alternative representation. In this model, the two lines representing the **RESPONDED EXISTENCE** and **EXISTENCE** constraints are combined into one line representing the **CO-EXISTENCE** constraint.

Because of its conciseness, the second model may be preferred over the first one. It delivers the information that the three activities always occur together in explicitly. The imperative model (Figure 3.10) implicitly contains the same information because without all three activities in one case, successful execution of the process from start to end is not possible.

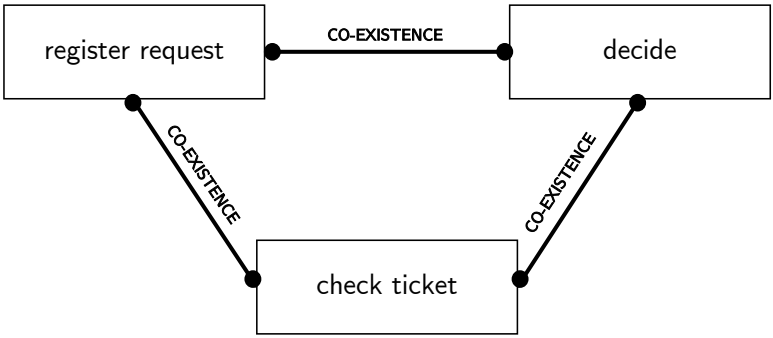


Figure 6.2: CO-EXISTENCE Representation of Association Rules for Compensation Request Process.

Inclusion of Attributes. The models in Figure 6.1 and 6.2 can be enriched with resource and costs attributes. Table 6.4 shows an excerpt of the association rules for the compensation request process when including the *resource* attribute into the frequent itemset and rule generation operators and setting the minimum support to 0.6. The process finds over 100 association rules whereby most of them have a support of one and contain a *decide/Sara* item indicating that *Sara* always decides about the compensation request. Other rules, including resource *Mike*, only have a support and confidence value of 0.667 while the resource is part of the consequent. Therefore, the information will most likely not be included in the declarative model. Figure 6.3 enriches the model in Figure 6.2 with resource information whereby *check ticket* is the activation and *decide* the

target activity for the **CO-EXISTENCE** constraint between them. The model could be extended further by including the *Costs* attribute. As the significance of association rules and their Declare representation is relatively limited, the following demonstrates the application of sequential pattern mining.

Table 6.4: Association Rules for Compensation Request Process (w/ Resource Attributes).

Premises	Conclusion	Supp.	Conf.
check ticket, register request	decide/Sara	1.0	1.0
check ticket, decide	decide/Sara	1.0	1.0
register request, decide	decide/Sara	1.0	1.0
...	...	...	...
decide, register request	check ticket/Mike	0.667	0.667
decide	check ticket/Mike	0.667	0.667
decide	check ticket, check ticket/Mike	0.667	0.667
...	...	...	...

Sequential Pattern Mining

Applying association rule mining on event logs can only find relations of activities without information about temporal orders. Table 6.5 shows the resulting patterns when applying the GSP algorithm with a minimal support of 0.6, a window size of zero, a maximum gap of Infinity, and a minimum gap of zero. In this case, the only input for the GSP operator is the set of activities in binominal format without any additional attributes. The process finds seven patterns whereby four of them have a support of one and the other three a support of $\frac{2}{3}$. Five patterns consist of two

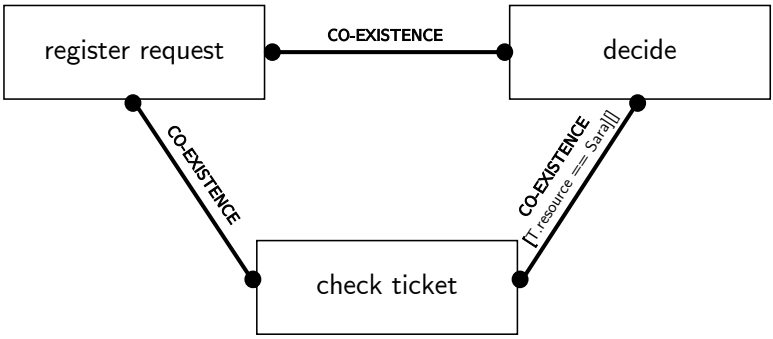


Figure 6.3: CO-EXISTENCE Representation of Association Rules for Compensation Request Process (w/ Resource Attributes).

transactions and items. Two patterns consist of three transactions and items. Figure 6.4 shows a translation of the sequential patterns to **RESPONSE** constraints in the Declare notation. This declarative process model contains one activity more (*examine casually*) when compared to the translation of the association rules in Figure 6.1 or 6.2. It expresses that whenever someone registers a request, there has to be a check of the ticket, a casual examination, and a decision. Moreover, a casual examination and ticket check necessarily lead to a decision.

Modelers should exercise restraint regarding translating sequential patterns with a support lower than one. Introducing a **RESPONSE** constraint between *examine casually* and *decide* is reasonable because whenever the request is examined (thoroughly or casually), there has to be a decision about the ticket. Conversely, for the **RESPONSE** constraint between *register request* and *examine casually*, the situation is different. Registering a request does not always lead to a casual examination. In $\frac{1}{3}$of the cases, a request is examined thoroughly. Therefore, this connection of the activities in the declarative process model may lead to misunderstandings and

Table 6.5: Sequential Patterns for Compensation Request Process.

Support	Transactions	Items	T1	T2	T3
0.667	2	2	register request	examine casually	
0.667	2	2	examine casually	decide	
0.667	3	3	register request	examine casually	decide
1	2	2	register request	check ticket	
1	2	2	register request	decide	
1	2	2	check ticket	decide	
1	3	3	register request	check ticket	decide

wrong interpretations. The support value of association rules or sequential patterns is crucial for deciding about their translation to Declare.

Increasing the minimum support value of the GSP algorithm to 0.8 removes three sequential patterns from the result set (all those that include the *examine casually* activity). The resulting process model in Declare notation looks exactly like the one in Figure 6.1 with the difference that the **CO-EXISTENCE** connections are replaced by **RESPONSE** types. Other parameter variations for the GSP operator are not useful for this dataset. In the process context, the window size should usually be zero because activities can happen one millisecond after another, and they should be treated independently. For vast datasets, a minimum or maximum gap variation may be useful.

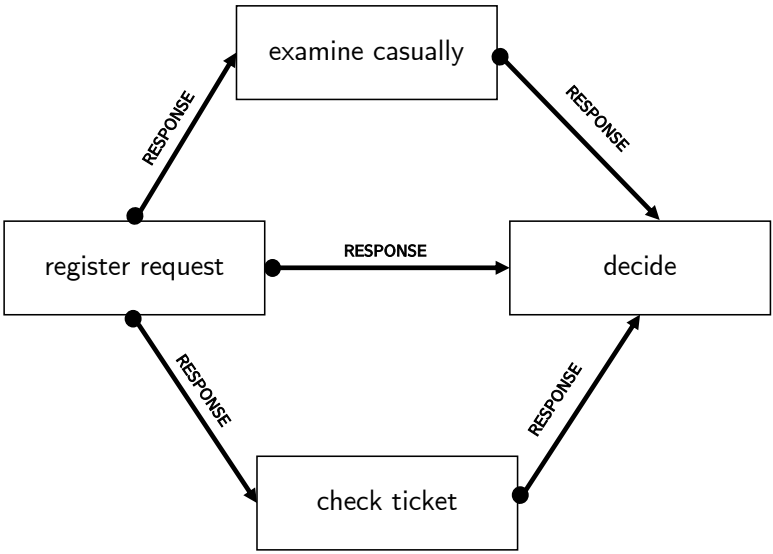


Figure 6.4: Declarative Representation of Sequential Patterns for Compensation Request Process.

Besides the complete view on the activities only, sequential pattern mining can also take into account the optional attributes, *Resource* and *Costs* in case of the event log used here. Beforehand, the *costs* attribute has to be discretized: Costs are *very low* up to 50, *low* up to 100, and *medium* up to 200. They are *high* above a value of 200. Setting the minimum support to 0.9 leads to a result set of more than 300 patterns. Table 6.6 provides an excerpt of the result set when including both the resource and costs information in the GSP operator. The output should be sorted so that larger patterns with longer sequences and more items (activities, resources, costs) appear on top of the result set so modelers can primarily translate them to Declare elements.

All patterns have a support value of one; thus, any one can be selected for transformation without the risk of constructing a model that describes infrequent process states or non-accurate behavior for many cases. Patterns including activities can be translated to **RESPONSE** constraints, analogously to the former sample without the resource and cost information. Note that both attributes are event attributes, i.e., they are related to a single activity and not the whole process instance. Therefore, patterns with transactions that only consist of resource or costs information have only subordinate importance for describing a process.

For instance, the pattern *{check ticket, Sara}* indicates that whenever the *check ticket* activity is executed, the value "Sara" is assigned as a resource for another activity afterwards. The (multi-perspective) standard set of Declare constraints has no matching constraint template for this pattern, though the information may be helpful for domain experts. For this reason, Chapter 5 has introduced **RESPONDED EXISTENCE** and **RESPONSE** constraints with indefinite activities that nevertheless can be addressed by an activation or correlation condition that involves attributes.

Single transactions with activity and resource information can be transformed to **EXISTENCE** templates of the respective activities. Consecutive combinations form activation or correlation conditions between two activities. Table 6.6 presents various patterns indicating that activity *register request* is eventually followed by *check ticket* and *decide* and another undefined activity. The first two activities are associated with very low or low costs, whereas the latter have medium costs. Resource *Sara* is associated with the *decide* activity. In sum, these types of patterns can form a **RESPONSE** construct in extended Declare notation like introduced in Chapter 5.

Another type of pattern includes the single resource *Mike* without any connection to an activity. An investigation of the event log reveals the reason for that: After registering the request, *Mike* is either responsible

for a casual examination or check of the ticket. Thus, a sequential pattern with *Mike* connected to one of these activities does not fulfill the minimum support of 0.9. Theoretically, these patterns can also be translated to a **RESPONSE** construct. However, as the first type of patterns contains one activity more (*check ticket*) and the information of the cost, it should be preferred over the patterns with resource *Mike*. Figure 6.5 shows a model construct based on the pattern *{register request, check ticket}* → *{decide, Activity D}* in Table 6.6, which has a confidence value of one whereby the last activity in the sequential order is undefined. The attributes in combination with activities are represented as an activation condition of the whole construct. The following discusses the results of applying association rule and sequential pattern mining on the compensation request process dataset.

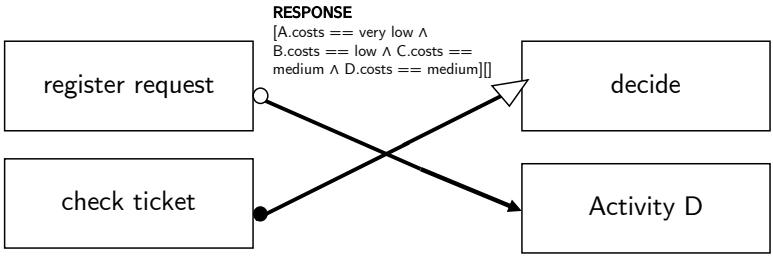


Figure 6.5: Declarative Representation of Sequential Patterns for Compensation Request Process with Resource Information (based on Table 6.6).

Results Discussion

Applying association rule and sequential pattern mining on this very small demonstrates the approach's applicability and verifies that the RapidMiner processes produce correct results. By comparing the declarative models

Table 6.6: Sequential Patterns for Compensation Request Process Including Resource and Costs Information.

Supp.	T1	T2	T3	T4
1	register request/very low	check ticket/low	decide/medium	medium
1	register request/very low	check ticket/low	Sara/medium	medium
...	...	...	...	...
1	register request/very low	decide/Sara/ medium	medium	
1	...	...	...	...
1	register request/very low	Mike	decide/medium	medium
1	register request/very low	Mike	decide/Sara	
...	...	...	...	...
1	decide/Sara	...	...	...
...	...	...	...	...

with the discovered imperative model in Disco (Figure 3.10 and 3.11), one can see that declarative representations provide nearly no understandability advantages. Quite the contrary, they do not contain the same information, and it is harder to capture the properties of the process. The imperative model (see Figure 3.11 in Chapter 3) can display all information of the event log and, humans can capture their meaning very well.

Still, the inclusion of *resource* and *costs* attributes in the sequential patterns is beneficial as it provides information about their relations to the activities at first sight. However, with the attributes included, the result set of the GSP algorithm gets significantly larger (more than 300 patterns even for high support values), which complicates the transformation process. It includes, as shown in part in Table 6.6, many patterns that represent the same information with for instance one activity or attribute more or less (cf. line 1 and 3 of Table 6.6). Their combined transformation to a Declare model element is relatively straightforward; however, it may overstrain employees who are not familiar with the data mining output.

Applying association rule and sequential pattern mining on the compensation request process supports the general hypothesis that declarative process models can cause some understandability problems [FLM+09]. More specifically, it provides evidence for the assumption that the declarative paradigm can play out its advantages better for larger and more complex processes. For the compensation request event log, association rule mining provides no significant advantages in addition to sequential pattern mining. The evaluation continues with a more extensive production process with more than 5000 events.

6.2.2 Production Process

Table 6.7 presents the key facts of the production process event log. It is derived from a data repository of the *Eindhoven University of Technology* [20] and is considerably larger than the compensation request dataset. The dataset contains five additional event attributes suitable for applying the data mining techniques in this thesis and process mining in general. Each activity is associated with a *Resource* attribute that indicates the machine on which the particular production step was executed. Attribute *Part Desc.* determines the product part that was manufactured in one

Table 6.7: Key Facts of Production Process [20].

Domain	Production
#Activities	27
#Cases	225
#Activity Executions	4543
Optional Attributes	5 (Resource, Part Desc., Worker ID, Report Type, Qty Completed)

run of the production process. It remains the same for every activity of one instance, i.e., case of the process, but several cases produce the same part. The *Worker ID* is comparable to the *resource* attribute of the compensation request process and assigns an (anonymized) worker to each activity. There are three *Report Types* (B, D, and S) whose exact meanings are not specified. *Report Type* D is by far the most frequent one (83 %), followed by S (16%) and B (1%), which leads to the presumption that D represents a success, S a review, and B a failed status. Finally, the *Qty Completed* attribute indicates an output quantity produced by a production step. Originally, the data originates from an export of an ERP system and describes a production process in the industry.

Process Mining in Disco

Figure 6.6 depicts the output of process discovery in Disco with standard settings (100% of the activities and minimum amount of paths shown). Applying process mining on this event log leads to a large process model with 27 activities and many paths between them. Users have a hard time understanding and capturing the process but can still manage to do so by thoroughly examining the process model. In particular, they can identify the happy path of the process. After *Turning & Milling*, the result of the

production step is checked (*Turning & Milling Q.C.*). Afterwards, the laser is set up (*Laser Marking*) so that lapping (*Lapping*) and grinding (*Round Grinding*) machines can perform their work. It follows a quality control (activity *Final Inspection Q.C.*) and the product is packaged (*Packing*). At last, the product package goes through a final inspection again for most cases. Activities that are not in the happy path perform setup, rework, or additional quality checks.

The 225 cases have 217 variants, i.e., 217 different process flows from the first to the last activity. By inspecting the variants more detailedly, users find out that 211 variants only include one case (i.e., execution of the process), two variants include three cases, and four variants consist of two cases. Setting the paths in Disco to 67% while leaving the number of activities at maximum makes the process model even more confusing, as shown in Figure 6.7. Now, the process model changes to a "spaghetti-like" diagram. The main problem is many paths between the activities, making it almost impossible to grasp the process flow. The resulting model is confusing for users with the maximum number of path setting.

Again, the hope is that a declarative modeling style can help get an overview of the most crucial relations of the process. As it did not deliver substantial benefit for the compensation request process, association rule mining results on this production process are omitted. Instead, the following presents the patterns resulting from sequential pattern mining right away. Like before, the resulting patterns are combined to transform them to a declarative representation in (multi-perspective) Declare notation.

Sequential Pattern Mining

The RapidMiner process for applying sequential pattern mining on the production process event generally stays the same as for the small compensation request process except for some minor additional preprocessing steps. Both *Activity* and *Resource* contain the respective other informa-

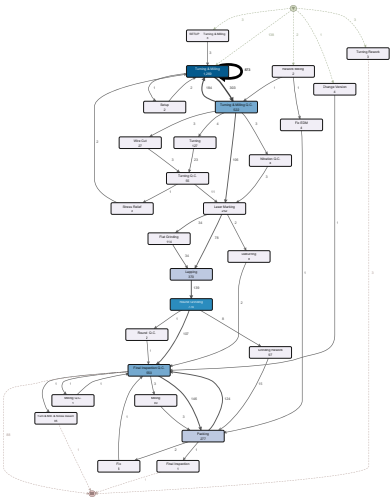


Figure 6.6: Discovery of Production Process in Disco [4].

tion concatenated with a hypen, e.g., *Turning & Milling - Machine 4* and *Machine 4 - Turning & Milling*. Therefore, the attribute must be split so that either the activity or resource information remains. Table 6.8 shows the result of the GSP process with no additional attributes included and a minimum support value of 0.6. The patterns include five different activities. As the support value is significantly lower than one, the sequential patterns must be translated into sequential rules. Their confidence values determine whether they are eligible to be included in a declarative representation.

Table 6.9 provides the confidence values for the rule representations of the sequential patterns in Table 6.8. Based on these rules, a declarative model that consists of a set of **RESPONSE** constraints can be constructed.

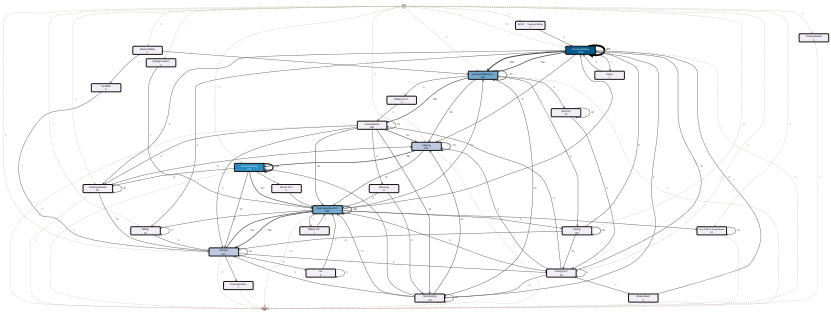


Figure 6.7: Discovery of Production Process in Disco [4] with Maximum Number of Activities and 67% of Paths.

The support values are relatively low, meaning that not all activities of the patterns or rules are present for a larger number of cases, however; the confidence values are still on an acceptable level. Therefore, a constraint resulting from the patterns is satisfied in many cases if the activating activity is part of the process instance. For these reasons, all patterns in Table 6.8 can be part of a transformation to a Declare model like shown in Figure 6.8.

The model concisely demonstrates the sequential dependencies derived from the happy path and summarizes them into one representation. Users should keep in mind that it does not show an absolute order of the activities. The packing could theoretically, even though this does not make sense for the real-world process, happen before laser marking as long as the **RESPONSE** constraint between *Laser Marking* and *Packing* holds. Another point to consider is that the activities are not present in every process case. Even if they are, the constraint does not always hold due to support and confidence values lower than one. Therefore, the declarative model should not be seen as a one-to-one connection to the event log.

Table 6.8: Sequential Patterns for Production Process (No Optional Attributes).

Support	T1	T2	T3
0.68	Turning & Milling	Turning & Milling	
0.649	Turning & Milling	Turning & Milling Q.C.	
0.644	Turning & Milling	Laser Marking	
0.636	Laser Marking	Final Inspection Q.C.	
0.631	Turning & Milling	Turning & Milling	Turning & Milling
0.622	Turning & Milling	Turning & Milling	Turning & Milling Q.C.
0.613	Turning & Milling	Laser Marking	
0.6	Laser Marking	Packing	

Here, including five optional attributes in the GSP algorithm is even more interesting than for the compensation request process. When providing the *Resource* attribute as an additional input and setting the minimum support to 0.7, it produces 15 sequential patterns whereby their maximum sequence length is two, and none of them includes an activity in both transactions of the pattern.

Examples are *{Laser Marking/Machine 7}* (support equals 0.742), *{Turning & Milling, Quality Check 1}* (support equals 0.707, *Quality Check 1* is a resource), or *{Quality Check 1, Laser Marking/Machine 7}*. It is not useful to construct a Declare model from this output. Therefore, the next step is to provide all four attributes as an input for the GSP operator.

Table 6.9: Sequential Rules and Confidence Values (based on Table 6.8).

Support	Sequential Rule	Confidence
0.68	Turning & Milling → Turning & Milling	$\frac{0.68}{0.72}$ = 0.94
0.649	Turning & Milling → Turning & Milling Q.C.	$\frac{0.649}{0.72}$ = 0.9
0.644	Turning & Milling → Laser Marking	$\frac{0.644}{0.72}$ = 0.89
0.636	Laser Marking → Final Inspection	$\frac{0.636}{0.74}$ = 0.86
0.631	Turning & Milling → Turning & Milling, Turning & Milling	$\frac{0.631}{0.72}$ = 0.88
0.622	Turning & Milling → Turning & Milling, Turning & Milling Q.C.	$\frac{0.622}{0.72}$ = 0.86
0.613	Turning & Milling → Laser Marking	$\frac{0.613}{0.72}$ = 0.85
0.6	Laser Marking → Packing Packing	$\frac{0.6}{0.74}$ = 0.81

To do so, the *Qty Completed* attribute is discretized as follows: Class *zero* is assigned to activities that produce zero products. Production quantities are *low* up to 50 and *medium* up to 100 products, everything above is considered as *high*. With minimum support set to 0.6 and every attribute except *Qty Completed* included, the RapidMiner process already runs for approximately 90 seconds (i7-7700 CPU, 3.6 GHz) and produces over 2000 patterns. Raising the support to 0.7 significantly reduces the runtime and number of patterns, but over 300 of them remain. With these settings, attributes *Worker ID* and *Part Desc.* do not appear in any of the patterns; thus, they can be excluded from the GSP operator. They do not play any role in finding sequential patterns with decent support values.

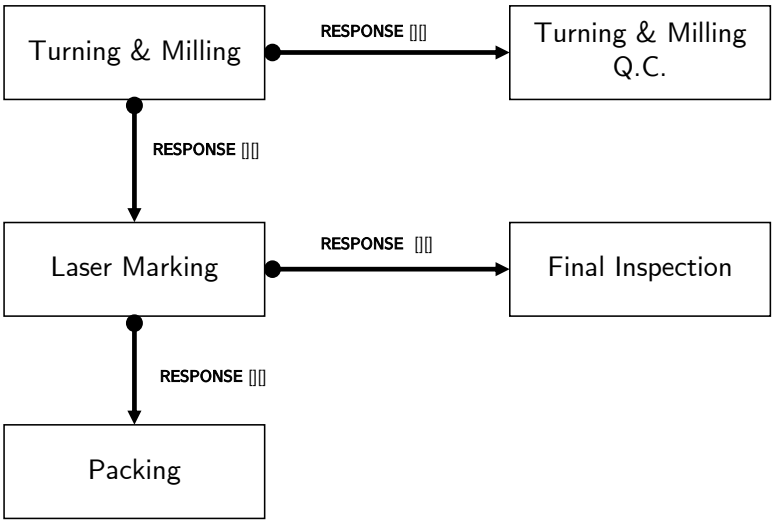


Figure 6.8: Declare Representation of Sequential Patterns in Table 6.8.

Conversely, the *Report Type* attribute is part of a multitude of patterns. Table 6.10 shows a small excerpt of the result set. The *R* in brackets after *Quality Check 1* clarifies that it is a resource attribute and not an activity. The *D* value originates from the *Report Type* attribute. When users take the output in Table 6.10 in isolation, it is not possible to produce a standalone declarative model. The reason is that the RapidMiner process does not find any patterns where a transaction that contains an activity follows another one of the same type. However, the resulting patterns could enrich the model depicted in Figure 6.8. Overall, the support values of the patterns with values of the *Resource* and *Report Type* attributes are even higher than for those with activities only. They could be introduced as activation conditions of the **RESPONSE** constraints even though they are not combined originally with the patterns in Table 6.8. Modelers should,

Table 6.10: Sequential Patterns for Production Process (Four Optional Attributes).

Support	Transaction 1	Transaction 2	Transaction 3
0.858	Quality Check 1 (R)	Quality Check 1 (R)	
0.782	Final Inspection Q.C./Quality Check 1		
0.742	Laser Marking	Machine 7	
...	...	...	...
0.778	Packing/D		
0.747	Turning & Milling Q.C./D		
...	...	...	...
0.742	Laser Marking/Machine 7/D		
0.724	D	D	Packing

however, consider that even though the support values are higher, there can still be cases in which the activation condition prevents an activation of the **RESPONSE** constraint that holds for this particular instance of the process. Figure 6.9 makes a proposal for a joined representation of both result sets.

Adding the *Qty Completed* attribute as an input dramatically increases the runtime of the process (several hours). However, the attribute does not appear in combination with activities. Instead, there are sequential patterns of the form *{Quality Check 1, low}*, *{Turning & Milling, low}*, *{low, D}*, or *{zero, low}* whereby *zero* and *low* are discretized values of the *Qty Completed* attribute. Moreover, these sequential patterns cannot usefully be included in the Declare model depicted in Figure 6.9 as connections to

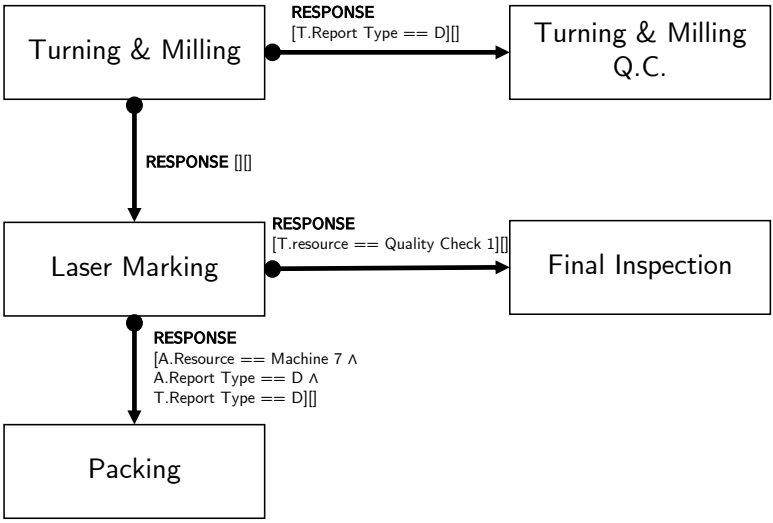


Figure 6.9: Declare Representation of Sequential Patterns in Table 6.8 Enriched with Results in Table 6.10.

the activities are missing. The following discusses the overall results of applying sequential pattern mining to the production process event log.

Results Discussion

In sum, applying the approach of this thesis on the production process event log seems to come with promising benefits. The output shown in Table 6.8 delivers a fast understanding of the essential activities of a process (part of the happy path) and their sequential dependencies. Process mining in traditional tools can also provide this information, but it comes with more effort for a manual investigation of the somehow confusing process models in Figures 6.6 and 6.7. Including the five optional attributes requires some additional preprocessing (discretization and splitting) of

attributes. Here, the challenge is finding the optimal support value that delivers meaningful patterns while keeping their number adequate. Adding optional attributes to the GSP operator and setting too low support values quickly leads to rapid growth, whereby many of them have solid commonalities and differ from others only in single or few parts. This could confuse and overstrain the users because they require effort to combine them into Declare model elements.

In general, combining two result sets like in Tables 6.8 and 6.10 to the model in Figure 6.9 should, for stated reasons, be accompanied critically. The output shows that the *Part Desc.* and *Worker ID* attributes are not included in the pattern; therefore, they can be excluded from the process. In this way, sequential pattern mining on the production process event log delivers information about which attributes commonly appear with which other attributes and which do not. These attributes are candidates for a further, more detailed investigation to identify performance improvement potentials or critical parts leading to bottlenecks. For the sample process here, for instance, this could mean that the company puts a special focus on the maintenance of *Machine 7* because downtime could cause severe problems for the whole production.

Process mining tools like Disco [4] also provide statistics about the process variants and the optional attributes (e.g., their relative frequency) but cannot establish connections between attributes and activities. While Disco can determine and analyze full process variants, sequential patterns usually shed light on parts of process executions. The production process event log contains data from three months. Process analysis for one quarter is already a realistic and practical scenario, but typically, unfiltered exports from essential information systems of a company like ERP, CRM, or production-related systems can be much larger. Therefore, the following evaluates the application of this thesis' approach on a vast, real-world ERP system export of a purchase order process.

6.2.3 Purchase Order Process of Production Firm

Table 6.11: Key Facts of Purchase Order Process.

Domain	Purchase Order
#Activities	42
#Cases	251,734
#Activity Executions	1,595,923
Optional Attributes	3 (Resource, Vendor, Document Type)

Table 6.11 presents the key facts of the purchase order process that was already touched in the motivating example in Chapter 1. The dataset originated from a real-world process of a multi-national Dutch company and was used for the BPI Challenge in 2019 [18]. First, it has to be transformed from the XES [Gro16] to a table-based format to be stored in RapdMiner. Van der Aalst et al. [vdABvZ17, 21] have developed RapidProM, a RapidMiner extension that brings functionality from ProM [7] to RapidMiner. Applying the *Import Event Log* operator allows importing an event log in XES format into RapidMiner. Afterwards, it can be stored as a dataset that eventually can be used like the compensation request and production process event logs.

The event log contains several additional attributes, whereby three of them seem to be suitable for being included in process mining activities. Similar to the previous event logs, the *Resource* attribute specifies either a human user or a batch job that executed one step of the process. For each case, there is a *Vendor* from which the company purchases the products. This attribute is a case attribute as it is the same for every activity of one process instance. At last, another case attribute *Document Type* specifies the type of purchase order. The overwhelming majority of

activity executions are associated with *Standard PO*s (over 1.5 million), the share of the other two values *EC Purchase Order* and *Framework Order* is under 5%.

Process Mining in Disco

Importing the dataset in Disco is straightforward because the tool automatically reads out all meta-information indicating which attributes represent which roles (e.g., case IDs, timestamps, other attributes) directly from the file in XES format. The process discovery results in the model depicted in Figure 6.10. Case and variants inspection in Disco finds that the most frequent variant of the process has a share of 20% of the cases. First, the company creates a purchase order item for which a selected vendor creates an invoice. Next, goods and invoice receipts are recorded and stored in the ERP system. Finally, the purchase order case is closed and archived (activity *Clear Invoice*). This sequence of events can also be derived from the process model in Figure 6.10 because the activities appear with a blue background color. In contrast to Figure 1.1 in Chapter 1 it is less confusing because it includes fewer less frequent paths between the activities.

Sequential Pattern Mining

Applying sequential pattern mining on the purchase order process with no further attributes included and a minimum support of 0.6 results in a runtime of approximately 15 seconds and 18 sequential patterns. This proves that the approach of this thesis is still applicable even for an event log with over one million activity executions. Table 6.12 presents an excerpt of the output for a minimum support of 0.7, which consists of sixteen sequential patterns in total. They can be transformed into a set of **RESPONSE** constraints in Declare. Such transformations can result in different model representations, one of which is shown in Figure 6.11.

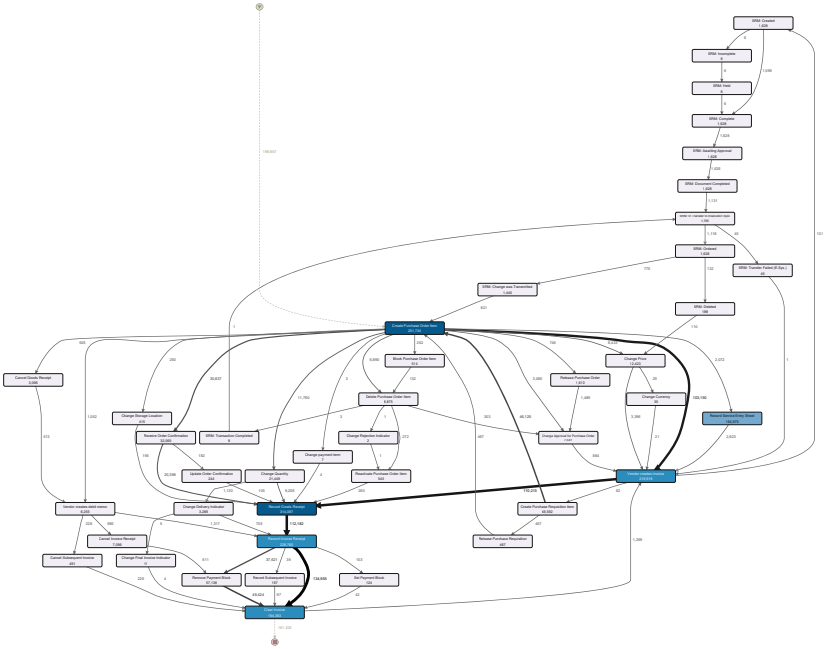


Figure 6.10: Discovery of Purchase Order Process in Disco [4].

Since the confidence values are relatively high, the choice of a particular representation is user-dependent and may vary given the actual context in practice.

When providing an optional attribute as an input of the GSP operator, the RapidMiner process runs out of memory (8 GB available) and terminates. Like in many comparable cases, one could apply the "kill-it-with-iron" principle and increase the hardware resources or run the process on high-performance servers. However, it is also possible to just

Table 6.12: Sequential Patterns for Purchase Order Process (No Optional Attributes).

Support	Transaction 1	Transaction 2	Transaction 3
0.926	Create Purchase Order Item	Record Goods Receipt	
0.84	Create Purchase Order Item	Record Invoice Receipt	
0.812	Create Purchase Order Item	Vendor Creates Invoice	Record Invoice Receipt
0.775	Record Goods Receipt	Record Invoice Receipt	
0.728	Vendor Creates Invoice	Clear Invoice	
0.724	Create Purchase Order Item	Record Goods Receipt	Clear Invoice

use less data by splitting it up into samples. RapidMiner supports various sampling strategies. Randomized approaches are not possible because the process data has a sequential order that must not be broken up. The *Filter Example Range* operator delivers what is needed here. It has two parameters that allow the users to specify a range of rows that should be kept. For the purchase order process, the range [1; 200,000] is chosen in a first approach, which includes the process data from approximately two months. The more performance a system on which the RapidMiner process runs has, the more data can be processed at once. When applying sequential pattern mining on such large datasets, analysts should ensure

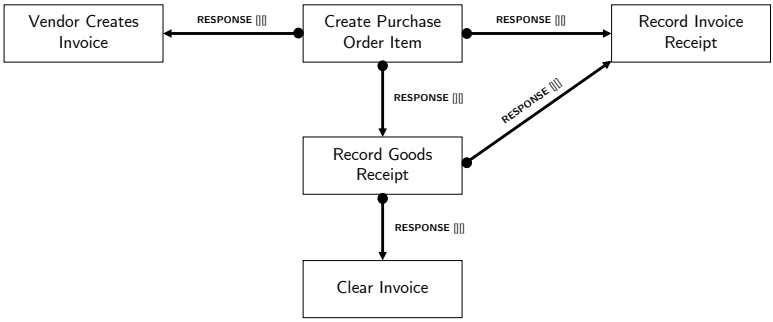


Figure 6.11: Declare Representation of Sequential Patterns in Table 6.12.

that every row of the entire dataset is processed at least once to avoid detecting process changes over time.

There should be more runs for the purchase order process with data of the ranges [200,001; 400,000], [400,001; 600,000], and so forth. When including only the *Resource* attribute, the runtime is about 30 seconds for a minimum support of 0.6 and 15 seconds for a minimum support of 0.7. A run with all three optional attributes and a support setting of 0.8 takes around two minutes and generates over 500 patterns. Table 6.13 presents an excerpt of the output. The *Vendor* attribute is not part of any pattern. Figure 6.12 shows one possible transformation to a multi-perspective Declare model. Confidence values are as follows: For instance, the rule *{Create Purchase Order Item/Standard PO}* $\rightarrow$ *{Record Invoice Receipt}* has a confidence of $\frac{0.859}{0.98}$ = 0.88. The rule *{Record Goods Receipt}* $\rightarrow$ *{Clear Invoice/Standard}* has a confidence of $\frac{0.847}{0.93}$ = 0.91.

One major difference to all previous Declare models based on sequential patterns from the sample datasets is that it includes a case attribute. The *Standard PO* value originating from the *Document Type* attribute states that standard purchase orders require a purchase order item, an

invoice by the vendor, and the recording of goods and invoice receipts. Creating an invoice for standard purchase orders is often not associated with any resource. Process or domain experts can decide whether this is expected and intended because the invoice creation is an outside activity or there should still be a resource assigned. In contrast to the production process (Figure 6.9), a full Declare model can be constructed from the sequential patterns with attributes alone; a combination with a Declare model without any activation or correlation conditions is not necessary.

Table 6.13: Sequential Patterns for Purchase Order Process (Three Optional Attributes).

Support	Transaction 1	Transaction 2	Transaction 3
0.859	Standard PO	Record Goods Receipt	
0.836	Standard PO	Vendor Creates Invoice	
0.728	Vendor Creates Invoice	Clear Invoice	
0.859	Create Purchase Order Item/Standard PO	Record Invoice Receipt	
0.847	Vendor Creates Invoice/ Standard PO/NONE		
0.847	Record Goods Receipt	Clear Invoice/Standard PO	
...	...	...	...

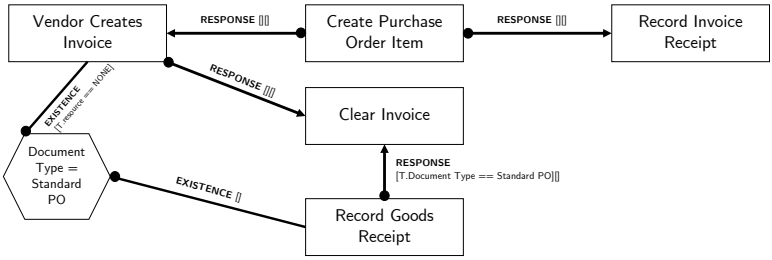


Figure 6.12: Declare Representation of Sequential Patterns in Table 6.13.

Results Discussion

Applying the RapidMiner sequential pattern mining algorithm on the purchase order process of a multi-national production company has proven that the approach of this thesis also works for large event logs, which commonly appear in practice. Furthermore, the GSP algorithm delivers patterns with reasonable support values that can be transformed to meaningful Declare constraints. It was shown that the approach works with case attributes. In practice, such attributes could even deliver the most interesting results because they apply for the whole process instances instead of only single events. The RapidMiner process finds that the attribute *Vendor* does not play any role in explaining the processes' coherences. However, like for the other two datasets, the *Resource* attribute seems to be closely connected to some activities.

After applying this thesis's approach to three diverse sample event logs, the following continues with a discussion about the overall findings and insights. This is accompanied by an evaluation of the general research questions introduced in Chapter 1.

6.3 Overall Evaluation Results Discussion

The application on the three sample datasets has provided a general implementation for association rule and sequential pattern mining on process event logs. Process analysts only have to adapt some preprocessing steps depending on the shape of the attributes and can quickly run the RapidMiner process. With the RapidProM extension [21], this is even possible for datasets that come in the XES format. Chapter 5 has contributed an implementation with which process analysts or other employees of a company can apply association rule and sequential pattern mining to all event logs suitable for process mining. With the transformation approaches shown in Chapter 5, Chapter 6 puts them to practice. The applications deliver several findings.

Process analysts face a challenge in choosing the minimum support parameter of the GSP operator. For larger processes and lower support values, the size of the result set can quickly increase to hundreds of patterns so that users cannot directly identify those worth considering for translating them to Declare model elements. However, setting the support value higher removes some possibly larger, but interesting patterns that are worth to be represented in the declarative process model. The applications to the three sample sets indicate that the support value should usually be between 0.6 and 0.8. Process analysts should most likely test different values within this range to achieve an acceptable tradeoff. Moreover, an automated approach that creates one possible Declare model based on the GSP output could support users and relieve them from finding the most reasonable choice of patterns. Also, automating the confidence calculation could help to simplify the overall approach for users who are not familiar with process or data mining techniques.

Furthermore, the applications have shown that declarative process mining, in general, should probably be combined with traditional process

mining projects. One of the advantages of the approach in this thesis is that it can connect activities and several optional attributes and represent them in a multi-perspective Declare model. The Disco tool, for instance, provides statistics about the relative frequency or the place in a process execution for activities and attributes. It is, however, not possible to establish connections between attributes and attributes. Another reason to use imperative and declarative process mining in combination is that it helps to calculate confidence values of sequential rules. Disco allows to set up filters for events and attributes that only select cases in which, for instance, particular activities appear and attributes have specific values. Filtering for, for instance, *Create Purchase Order Item* and *Standard PO* leaves 98% of cases. Dividing the support of a sequential rule constructed from the pattern in line 4 of Table 6.13 by 0.98 leads to a confidence value of $\frac{0.859}{0.98} = 0.88$.

Applying the data mining techniques on three sample event logs has also shown that association rule mining is less useful than sequential pattern mining. Although it may be slightly easier to set up and understand the construction of the output, these advantages do not outweigh the lack of sequential order for the association rules it produces. Also, it does not deliver any crucial additional constraint types. With sequential pattern mining, modelers can introduce **EXISTENCE** constraints and **RESPONSE** as well, which should usually be preferred over **RESPONDED EXISTENCE** in the process context. Thus, whenever possible, process analysts can apply sequential pattern mining alone in RapidMiner or any data-aware programming language like Python without losing essential results. Only in exceptional cases, when the focus is on specific co-existence relations of activities, association rule mining may produce superior output.

The following Chapter 7 shows research related to the contribution of this thesis. The goal is to delineate the approach of this thesis to other

approaches for declarative process discovery in the literature and highlight its advantages, also for the multi-perspective case. Furthermore, the chapter relates the approach with general contributions in the declarative process mining and modeling research field.

Part III

Related Research and Conclusions

Part 3 ("Related Research and Conclusions") categorizes the approach of this thesis into the research context. Chapter 7 presents related research. Besides a differentiation to other approaches for declarative process discovery, it consists of an elaboration of research in the field of multi-perspective declarative modeling and mining as well as other related research in general.

Chapter 8 concludes the thesis. Thereby, it summarizes the main contributions and points out again the main benefits of combining data mining and process mining. Finally, it takes a look at possible future research interests. Examples for such interests could be improvements for the approach like an automatic transformation of the rules and patterns or a real-world study to evaluate its usefulness and user satisfaction based on practical experiences in the industry.

7 Related Work

This chapter encompasses related work and research starting with prior work on declarative process discovery that only considers the control flow perspective. Thereby, alternative algorithms, approaches, or processes that in some way are related to declarative process discovery based on business process event data are introduced. The goal is to delineate the approach of this thesis from others and point out its advantages and drawbacks. Next, the research area is extended to approaches that also take into account additional attributes. Section 7.2 presents various contributions for multi-perspective declarative process mining or modeling. At last, the chapter focuses on related research that does not contribute a concrete approach but, for instance, applies declarative process mining or modeling in practice. These experiences can be of prime importance to understand the benefits of the declarative paradigm.

7.1 Prior Work on Declarative Process Discovery

Declarative Discovery in ProM

Maggi et al. [MMvdA11] present an approach for the automatic discovery of declarative models (in Declare notation) from process event logs. Their application is implemented as a plugin in ProM [7]. The algorithm requires a set of user-defined Declare templates and an event log. Then, the approach generates a set of candidate constraints which are then translated to LTL formulas. Afterwards, it is checked whether the LTL rules hold

on the event log. Fulfilled rules are included in a Declare process model. To reduce the runtime of the algorithm, a parameter *Percentage of Events (PoE)* can be defined that makes the algorithms only generate a certain percentage of the most frequent event classes. Additionally, the *Percentage of Instances (POI)* parameter defines a certain minimum support value that is acceptable for constraints under which they still get discovered.

The authors also address the problem of truncated process logs, i.e., situations where only an excerpt of the examined process is available in an event log, and the problem of constraints that are vacuously (trivially) satisfied. They state that Apriori-like approaches [AS+94] cannot generate negative constraints and full model from an event log and apply their approach to a case study (vessel navigation process) in the maritime sector.

In contrast to the approach in this thesis, Maggi et al. generate a set of candidate constraints beforehand and then achieve a reduction by checking them on the event log. Also, the user involvement happens at the start of the generation process with the selection of the Declare template set. Users can influence the size of the result set through two parameters. One advantage is that the algorithm can also find the strengthened versions of the base constraints, e.g., **ALTERNATE RESPONSE** and those with negative semantics like **NOT CO-EXISTENCE**. Like many others, their approach is based on candidate set generation, which does not allow the user to retrace the introduction of a particular Declare constraint.

Maggi et.al [MBvdA12] continue their work with the development of an Apriori-based approach for declarative process discovery. Their second approach is two-fold: A set of candidate constraints is generated in the first step. Afterwards, this set is pruned by measuring confidence, support, and other more advanced metrics. In contrast to the approach presented in this thesis, however, the Apriori algorithm is not used for the actual generation of the Declare constraints but for the generation of frequent activity sets, leading to a set of candidate Declare constraints on them.

Thereby, the Apriori algorithm drastically reduces the search space by using the property that all non-empty subsets of a set are also frequent and all supersets of infrequent sets are also infrequent.

Like their first approach [MMvdA11], it can consider negative or non-occurring events. A post-processing procedure follows after the set of Declare constraints has been generated based on the frequent activity sets. The authors define support, confidence, interest factor, and conditional-probability increment ratio (CPIR). These metrics are used further to reduce the amount of interesting and fitting Declare constraints. The paper uses an insurance claim process and an excerpt from a request process in the Dutch municipality to evaluate the described techniques. Results show that the advanced approach of Maggi et al. significantly reduces the size and complexity of the resulting Declare model compared to the former naive approach. Still, the described approach is based on the generation of candidate constraints which are then reduced using pruning techniques. The found constraints are not directly justified contentwise based on sequential patterns or rules. This could lead to difficulties for users to comprehend why particular constraints have been included in the declarative process model, and others have not.

Westergaard et al. [WSR13] contribute another approach for the discovery of declarative models from event logs. It is implemented in ProM [7] as well. They state that their *UnconstrainedMiner* technique addresses the shortcomings of existing approaches. One limitation they identify is the unclear semantics of Declare constraints for finite traces. Therefore, the authors propose introducing regular expressions as new semantics for Declare constraints. Regular expressions can be expressed as finite automatons; hence, a transformation from Declare constraints to finite automatons is possible. The approach generates candidate sets by combining all process activities with all Declare constraints. For each log trace, the algorithm checks whether the finite automaton of the particular

constraints ends in an end state. In this case, the constraint is fulfilled for the process instance. Moreover, the authors criticize other declarative discovery approaches for making the user specify a particular set of constraint types, which the algorithm then searches for on the log.

The authors claim that their approach can discover all constraints and includes post-pruning techniques to reduce the number of constraints. It takes the ProM Declare Miner [Mag13] plugin as a starting point and applies four different techniques that improve the efficiency of the algorithm. Westergaard et al. claim that even their base implementation is faster than competing solutions, and improvements significantly enhance the performance. This is achieved by several pruning methods like symmetry reduction, prefix sharing, or parallelization. The technique is also based on the generation of candidate sets and shifts their checking to an intermediate automaton representation.

ProM-independent Implementations of Declarative Discovery

Di Ciccio and Mecella [DCM13] contribute an implementation of declarative process discovery outside ProM [7]. Their *Minerful*$^{++}$ algorithm is rather complex and requires the event data to be in string format, e.g., *bcaac* whereby *a*, *b*, *c* denote activities. The authors introduce a regular expression for each Declare constraint, for instance, *[^a]*(a.*b)*[^a]** for a **RESPONSE** constraint between activities *a* and *b*. *Minerful*$^{++}$ consists of two main phases. In the first phase, a knowledge base is constructed based on applying distance and appearance functions on the event log. Phase two discovers the declarative constraints. Functions that include the distance and appearance sets calculate the support values of potential constraints. The regular expression of the constraint determines the concrete shape of such a function it is assigned to. Di Ciccio and Mecella claim that their algorithm achieves results comparable to the approach by Maggi et al. [MBvdA12].

Declarative Discovery for Alternative Notations

Debois et al. [DHLU17] propose an algorithm for the generation of declarative process in the Dynamic Condition Response (DCR) graph notation. DCR graphs are a graph-based notation that describes the process through a set of activities, boolean values indicating the status of activities, and relationships between the activities [HM11]. An activity can have three different status, namely *executed*, *included*, and *pending*. Four templates (*response*, *condition*, *exclusion*, *inclusion*) define the relationship between activities. Starting from four well-known metrics (*fitness*, *precision*, *simplicity*, *generality*) the authors present the DCR mining algorithm.

The algorithm, like others, generates a set of candidate constraints and then checks whether they are fulfilled on the log or not. In this case, the start set of candidate constraints encompasses *condition*, *exclusion* and *response* relations. Afterwards, the algorithm performs a search space run by setting *included* states, checking for *response* fulfillments, and removing conditions according to certain principles described by the algorithm. Based on support and confidence measures, constraints are weighted and potentially removed from the candidate set. In a post-processing step, redundant constraints are removed. An application on a sample log and a comparison with the Declare Maps Miner [Mag13] shows that Debois et al.'s approach generates comparable output in DCR graph notation.

Declarative Discovery Based on Textual Input

Van der Aa et al. [vdADCLR19] propose an approach for the automatic generation of declarative process models from textual input. Textual input, in this case, means a description of the process created by a human, e.g., an employee who is part of the process or a consultant who has interviewed in a company. Their work represents one of the first contributions that outputs declarative model elements based on process descriptions instead of event logs. Several contributions result in imperative process models

such as Petri nets or BPMN. Thereby, the authors make use of well-known techniques of *Natural Language Processing (NLP)*. The approach focuses on five Declare constraints (**INIT**, **END**, **RESPONSE**, **PRECEDENCE**, **SUCCESSION**) and consists of three steps. In a linguistic preprocessing step, NLP techniques are applied to structure the sentences concerning the semantics of their single components, for instance, part-of-speech-tagging classifies verbs, subjects, objects as well as interrelations of a sentence. Afterwards, the activities and their names are extracted. Van der Aa et al. [vdABMN20] also make a first contribution for the use of speech recognition for constructing declarative models, Alman et al. [ABMvdA20] present a chatbot for the specification of Declare constraints.

In a final step, unary and binary Declare constraints are constructed based on the activities found and their semantic relationships. The authors evaluate their approach by comparing automatically found constraint patterns with manually created ones. Precision and recall values lead to an overall F1-score of 0.74. The crucial difference to the approach presented in this thesis is the input format (textual process descriptions instead of event logs). Also, it solely considers the control flow and no additional data perspectives. Like other declarative process discovery approaches, van der Aa et al. focus on extracting a limited set of Declare templates. The authors state that the extension of the approach to other, more sophisticated templates, like for instance **ALTERNATE PRECEDENCE**, is part of their future work [vdADCLR19].

7.2 Multi-perspective Declarative Process Modeling and Mining

Maggi et al. [MDGBM13] develop a technique for the discovery of declarative constraints from event logs that are enriched with data conditions,

representing them by using an extended Declare notation. Like the other approaches by Maggi et al. ([MMvdA11], [MBvdA12]), this data-aware declarative process mining approach is based on the generation of candidate constraints. In the next step, the constraints are squared with traces of the log, and it is determined whether and where an assignment of a data value holds. Afterwards, supervised learning techniques like decision trees are applied to the set of candidate constraints to find the right values for the data. Non-relevant candidates, selected based on a threshold of activation, are pruned. In a second step, the remaining constraints are evaluated regarding their violation or fulfillment on the log. Again, the user has to select a particular type of constraint that she or he is interested in (binary relation templates).

Burattin et al. [BMS16] contribute a technique for conformance checking using the declarative model paradigm and a multi-perspective extension of the Declare notation (MP-Declare). Their approach takes an event log and an MP-Declare model (as introduced in Chapter 2) as input and then iteratively checks the fulfillment of constraints on the traces. The authors illustrate their approach using the **RESPONSE**, **ALTERNATE RESPONSE** and **CHAIN RESPONSE** constraint template of Declare. The algorithms are implemented as a plugin for ProM [7]. Applications on three real-life case studies have shown their general operability.

Schönig et al. [SCJM15] propose an approach for the discovery of constraints that deal with resource assignments considering both workflow and data perspective, using the Process Intermediate Language (DPIL) [ZSJ14] as the process notation. DPILis an alternative to other declarative modeling languages like Declare or DCR graphs. It allows a textual description of activity, resource, and organizational unit relationships through a set of patterns. The *DpilMiner* relies on the principle of candidate sets generation as well and can output a set of constraints when applied to an event log, whereby pre- and post-processing techniques

reduce the number of candidate constraints that have to be checked. The authors evaluate their approach on a dataset of a business trip management system of a university, considering the organizational perspective alone and together with the control flow perspective.

They claim that their approach achieves results that are comparable to the DeclareMiner [Mag13]. In the case study, 85 percent of the rules found with resource bindings (one particular resource has to perform an activity), organizational bindings (some role has to perform an activity, e.g., supervisor), and role-sequence bindings (mandatory or recommended sequence of a process part depending on the role of the resource) were rated as relevant. In contrast to the approach of this thesis, Schönig et al. [SCJM15] use the DPIL notation for declarative process models and intensively focus on the organizational perspective or only a special additional attribute of the event log like role or resource. Competing multi-perspective declarative process mining approaches consider all kinds of attributes. However, the results have similar semantics when compared with the activation and correlation conditions of multi-perspective process models that are, for instance, used in RuM [ADCH$^+$].

Schönig et al. [SRSC$^+$16] develop a declarative discovery algorithm based on SQL queries, this time using the Declare notation [PSVdA07] as the output format. The event data is stored in a relational database using the RelationalXES format [vDS15]. A general query template is adapted depending on the constraint type (e.g., **RESPONSE**) and support and confidence desired. It also allows discovering strengthened versions of the base constraints, such as **CHAIN RESPONSE**.

Like for their approach using the DPIL notation, they address the challenge of including additional attributes into the constraints, whereby they take the *resource* attribute as an example for the organizational perspective. The authors go even one step further and explain how the query can be adapted to find role-based **RESPONSE** constraints. Such constraints

include a condition that activates it only in case a particular role is assigned to the activating activity. They closely match the representation of **RESPONSE** constraints with activation conditions on the activating activity in a Declare model like introduced and used in Chapters 5 and 6.

The SQL-based approach is evaluated on two real-life event logs. It performs slightly worse than the competing approaches of Di Ciccio and Mecella [DCM13] and Westergaard et al. [WSR13] but significantly better than the DeclareMiner by Maggi et al. [MBvdA12]. Schönig et al. state that the lower performance level is compensated through expressiveness and customizability of the SQL queries. The most apparent difference to the approach of this thesis is the storage format of the event data. Starting from logs in XES [Gro16] format, the approach requires an implementation that loads the data into a database so that it can be accessed with the SQL queries. Users must be fairly skilled and experienced in constructing SQL statements to extract the most out of the event database.

In another work, Schönig et al. [SDCMM16] present a generalized framework for discovering multi-perspective process models also taking other types of attributes into account. The multi-perspective extension of Declare describes four kinds of conditions, namely activation, target, correlation, and time conditions. Activation conditions have to hold to activate a particular constraint, i.e., to include it into the set of relevant declarative model elements for a specific trace of the process. Target conditions introduce value requirements for executing a particular activity, whereas correlation conditions define relationships between attribute values of two activities. Time conditions define time windows in which a second activity has to be executed after the first one.

The framework uses the Structured Query Language (SQL) and allows querying relational databases that are filled with event log data using the RXES format [vDS15] architecture. This is possible for standard declarative discovery and the multi-perspective case and results in overviews of

existing constraints with related attribute conditions as well as Support and Confidence values. The framework is evaluated using three event logs. Thereby the authors show the general applicability and the generation of the different kinds of constraints. Schönig et al.'s approach is one of few ones that uses SQL to extract declarative constraints from event data. One disadvantage is that a person who wants to generate constraints must be skilled in creating complex SQL statements, which may not be the case for persons with a non-technical background. Like before [SRSC+16], the RXES architecture is not a standard format for the storage of process event data. Usually, the event data is extracted from the information systems that support the business processes and transformed and integrated to an event log in a CSV, XES or any other XML-based file format.

Navarin et al. [NCB+20] present a technique for discovering multi-perspective declarative process models on streamed event data. In contrast to offline event data in the form of event logs, streaming event data is not a fixed set of data but is constantly changing and growing. Application areas are, for instance, the live monitoring of process executions. Application areas are for instance the live monitoring of process executions. The authors use Hoeffding decision trees suitable for the analysis of data streams. In the first step, rules are created using the Lossy Counting algorithm. Then, these rules and satisfying and violating events are the input for a Hoeffding tree structure that learns a classifier with data conditions. Finally, data conditions are derived from this model. Navarin et al.'s approach is limited to **RESPONSE** and **PRECEDENCE** Declare constraints. Their evaluation with datasets shows that if the budget for constructing the tree structure is high enough, the approach achieves results that are comparable to other offline discovery techniques like presented by Maggi et al. [MDGBM13].

Alman et al. [ADCH+, ADCM+21] present an application called RuM that implements various techniques for process mining with the declar-

ative paradigm. RuM implements declarative process discovery, conformance, and logs generation using the Declare notation. Both discovery and conformance work together with multi-perspective Declare models, including a data perspective. The log generation functionality allows the generation of an event log when providing a (multi-perspective) Declare model as input. A multi-perspective Declare editor provides the possibility to change the data-ware constraint types (activation, correlation, time) as well as to add, remove or edit attributes of the respective activities. This may be useful for cases when the multi-perspective discovery algorithm does not perfectly find the data-aware conditions or the user has more domain knowledge about the process and wants to add some information manually. In its total, RuM is one of the first tools that combines a large set of declarative process mining functionality. It implements various previously mentioned algorithms and approaches and presents them in a user-friendly graphical interface.

The declarative process mining approaches introduced in Section 7.1 do not consider additional perspectives, i.e., they do not include additional event log attributes into the constraints they produce. Some produce constraints in alternative notations (DPIL, DCR graphs) or require different input formats (e.g., event data relational database tables). Several techniques use candidate set generation and check whether they hold for the event log. Highly-complex algorithms make it impossible for users to retrace the reason for which a particular constraint has been introduced.

Things are similar for the multi-perspective modeling and mining approaches in Section 7.2. Schönig et al. [SDCMM16] contribute one of the first SQL-based approaches for discovering multi-perspective Declare constraints. Like their previous discovery approaches, it requires event data in RXES format [vDS15] and custom SQL queries created by a user. One significant advantage is that it generates constraints, including activation, correlation, and time conditions. The RuM application by Alman

et al. [ABMvdA20] provides a user-friendly environment for all kinds of declarative process activities. However, it is not clear which declarative mining algorithm it implements. Multi-perspective mining does not seem to be included. Section 7.3 introduces, inter alia, case study applications of declarative process mining. The goal is to derive insights into practical benefits, challenges, and implications for future research efforts.

7.3 Related Research for Declarative Process Modeling and Mining

Rojas et al. [RMGSC16] perform a literature review about process mining applications and case studies in healthcare, a typical application area for process mining in general. The result summarizes applications in healthcare in Germany and The Netherlands in oncology and surgery. Some aspects examined in the survey are the types of processes and data, the techniques or algorithms used, and the perspectives considered. One of the publications found is a work from Rovani et al. [RMDLVDA15] about the use of declarative process mining in healthcare. Although the doctors participating in the case study needed some time to grasp the Declare notation fully, the authors report successful conformance checking between normative models and the actual executions with benefits for the patient treatment process.

Pichler et al. [PWZ+11] perform an experimental comparison of imperative and declarative process modeling approaches. Students of a business management class participated in a study and were confronted with sequential and circumstantial tasks about sets of declarative and imperative process models. The Findings suggest that the imperative process models were understandable more easily. Although the results cannot be taken as face value because of the relatively small group of students and their prior

experience with imperative approaches, the study confirms the subjectively experienced tendency that declarative process models come with a certain difficulty for model understanding. Hence, it delivers another reason for research to develop ideas to tackle this challenge. This thesis delivers a contribution as it shows an approach for declarative process discovery with a transparent illustration of the patterns and rules (as well as their translation) leading to the declarative model constraints.

Debois et al. [DS15] contribute a report about the use of declarative process modeling notations in practice. Their paper has two main research questions. On the one hand, they evaluate whether practitioners using the declarative process modeling paradigm use its advantages and adapt their work routines. On the other hand, the authors assess whether it is always possible to construct a flow-based model that represents the same information as the set of declarative constraints. By analyzing the DCR graph model and the corresponding execution traces regarding variability, the authors conclude that the flexibility advantages of the declarative paradigm in fact have been used by the company.

The authors apply several implementations of process discovery algorithms on the event log and conclude that none of them can produce an appropriate imperative representation. However, this does not mean that such a flow-based model does not exist. The creation of hand-made models could be a topic for further research. Finally, Debois et al. refer to the field of hybrid process modeling notations where some parts of a process are modeled using declarative and others using imperative notations. Overall, the paper contributes a real-world application scenario in practice and shows that the advantages of declarative notations can be exploited in companies. Thus, the case study could serve as a role model or manual and guide other firms that plan to employ such techniques.

Prescher at al. [PDCM14] contribute a further investigation of the second research question of Debois et al. [DS15]. The authors develop

a framework for transforming Declare models to Petri nets. Their approach works in a three-step procedure, wherein a first step declarative constraints are translated to regular expressions that are then further transformed to finite state automatons that can be represented as Petri nets. The conclusion is that it is always possible to construct behaviorally equivalent Petri nets from a set of Declare constraints. Prescher et al. provide an implementation of their approach and evaluate it on a sample data set from the BPI challenge 2013 [8]. The finding is not necessarily in conflict with Debois et al.'s [DS15] paper because Prescher et al. do state anything about the resulting Petri net's generalization, precision, or compactness. In the broad research field, the paper plays a role in transforming declarative process models to imperative notations like Petri nets and vice versa and their interaction in hybrid business process representations.

All in all, the practical application and other related research contributions show that declarative notations can indeed be employed in practical scenarios. More research should be conducted, especially about their combination in hybrid process representations. After having introduced and discussed related work and research, Chapter 8 concludes this thesis and points out possible directions for future research.

8 Conclusions

This thesis has shown an approach for applying association rule and sequential pattern mining to process event logs. The following summarizes the main aspects and contributions and provides implications for practice. Next, Section 8.2 makes a first proposal for integrating process mining in general and the declarative discovery of this thesis in particular into the Horus method. It addresses the conceptual level and discusses the relationships between attributes of an event log, object stores (places) of the Petri net procedure model, and the data model. Finally, the chapter gives an outlook to possible future work and research interests, focusing on practical applications of the fundamental data mining techniques.

8.1 Summary and Implications

One of the main contributions of this thesis is a detailed manual for applying association and sequential pattern mining on event logs. Starting from event data with three mandatory attributes preprocessing steps prepare an event log for either frequent itemset generation or the GSP algorithm. This thesis uses RapidMiner as one representative of a GUI-based data mining and machine learning application. Both association rule and sequential pattern mining processes can be reused by importing other event logs and, if necessary, adapting the preprocessing steps. However, other tools that can deal with datasets like Knime [22] or SAS [23], or programming languages like R or Python are also suitable.

Chapter 5 shows several types of association rules and sequential that can result from both techniques and how they can be translated to some declarative constraints in the Declare notation. Moreover, the Declare language is extended to capture some of the rules and patterns better. In Chapter 6, both techniques are applied to three sample datasets from web repositories. The datasets differ in size and complexity of the process they describe. For all of them, running the RapidMiner processes is successful and leads to a (more extensive) set of rules or patterns.

Here, one of the challenges of the approach to appropriately set the minimum support and confidence parameters arises. Too low values let the techniques produce results of enormous size, too high values come with the danger of pruning interesting relations. Applying the approach on the sample datasets has shown that support values should usually be between 0.6 and 0.8. Additionally, it can be seen that in most cases, association rules do not deliver substantial benefits for constructing a declarative model in comparison to sequential patterns.

This thesis puts a special focus on the practical relevance of declarative process mining. The straightforward application of association rule and sequential pattern mining on event logs comes with several advantages and drawbacks. One of the drawbacks is that performance (time and space complexity) is directly connected to the size of the datasets. No further pruning or other optimization techniques are implemented apart from those included in Apriori-based frequent itemset generation or the GSP algorithm.

However, the sample applications have shown that the approach is still feasible even for very large event logs with more than one million events. There are two ways to deal with performance problems. Either the dataset is split into several periods (e.g., months) and processed successively or the hardware resources are upgraded ("kill-it-with-iron"). Since the RapidMiner processes only run for approximately one minute with 200,000

events on an ordinary personal computer, high-performance servers will most likely resolve all performance problems.

One of the advantages of the approach in this thesis is that it clearly demonstrates the transformation processes to modelers or any employee concerned with the declarative process discovery task. They can decide which constraints they want to include and base their decisions on process domain knowledge. This flexibility comes with the drawback that modelers must first have this domain knowledge and second must be trained in handling association rules, sequential rules, and declarative process models in general. Furthermore, additional attributes are included in the association rules and sequential patterns. They can directly be translated to multi-perspective Declare constraints as well, whereby the set of Declare templates is extended to match larger rules and patterns more closely.

Companies face a challenge integrating process mining into the established BPM life cycles. In an ideal world, process modeling and mining activities run side by side, and the results of both techniques are combined for optimal analysis results and actions derived from them. Not only companies for their internal processes, but also consultancies that offer process consulting have to adapt their portfolio to survive in competition with other firms. The following makes a first proposal to integrate process mining into the Horus method. A particular focus is put on how declarative process mining and the approach of this thesis can be combined with process modeling using the Horus notation and Horus Business Modeler [10].

8.2 Integration into the Horus Method

Figure 8.1 shows an adapted procedure model of the Horus method (cf. Figure 3.1 in Chapter 3). The *Prepare* and *Strategy & Architecture* phases

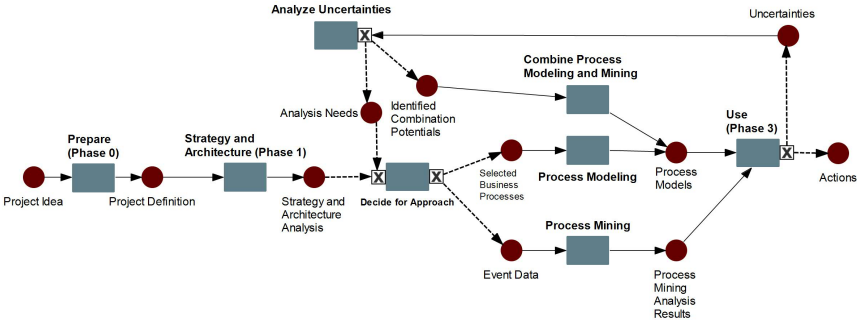


Figure 8.1: Procedure Model of the Horus Method Including Process Mining.

remain as before. For simplicity reasons, Figure 8.1 only depicts the *Procedure Analysis* part of the *Business Process Analysis* phase. After the first two phases, a decision must be made whether process modeling or mining shall be performed. The basis of this decision-making is, among other things, based on the availability of event data, the effort it takes to extract and transform them to a suitable event log, or the expertise of potential interviewees. In both cases, either hand-made process models or process analysis results (process models, KPI values) are input into the *Use* phase of the Horus method. If process management, implementation, or performance management can be executed properly, actions derived from the analyses are put into practice. If this is not possible, employees have to analyze the uncertainties and identify the analysis needs or combinations potentials between process modeling and mining results. In the former case, there has to be a decision to perform process modeling or mining again. Process mining and modeling results are combined if employees identify such potentials.

Enrichment of Horus Procedure Models

Process modeling results in models, usually in the form of a process graph like in Figure 8.1, whereby process mining discovery delivers models in either the imperative or declarative form like produced by the approach of this thesis. In the context of the Horus method, one of the goals of combining the techniques' results is to create a comprehensive procedure model. Horus allows specifying resources, roles, and costs for each activity. As seen in this thesis's previous chapter, this information is commonly part of an event log.

The approach of this thesis can provide the basis for multi-perspective Declare models where such attributes can be included in activation or correlation conditions. Therefore, the *resource*, *roles*, and *costs* property of an activity in Horus could be derived from a multi-perspective model. This has the advantage that these properties are set based on actual process executions. However, as the process models in Horus are usually normative, modelers should carefully decide whether the property values found by analyzing the event logs differ from their expectations or not. In this way, combining process mining and modeling results leads to conformance checking between the output of both.

Enrichment of Declarative Process Models

Enriching declarative models resulting from mining activities is possible as well. The approach of this thesis suffers from the drawback that the association rules and sequential patterns can only be translated to a limited set of Declare constraints, e.g., **RESPONSE**, **EXISTENCE**, or **RESPONDED EXISTENCE**. In particular, the introduction of strengthened versions of the base constraints like **CHAIN RESPONSE** or **ALTERNATE PRECEDENCE** is not possible. With domain knowledge about the process and a Horus model at hand, modelers can convert base constraints to the strengthened version if, for instance, they know that some activity must

necessarily appear directly after another or can only appear if another activity has appeared before without the activity itself in between.

Conformance Checking

The approach of this thesis can also contribute to conformance checking. Assuming that the declarative process models are normative, modelers can check whether the declarative constraints hold on the Horus procedure model. Consider a **RESPONSE** constraint between two activities. An execution path must eventually always lead to the constraint's target activity. Note that procedural models could also offer other paths that do not come across the second activity of the constraint without misdescribing the process. However, depending on support and confidence values, such constraints could trigger a rethinking of the Horus model. Modelers may rearrange parts of the model so that the constraints are satisfied in more cases or, for instance, delete paths that could violate them.

8.3 Outlook

The thesis leaves some questions that can inspire future research efforts, especially regarding the practical implications of the approach and its integration in current BPM structures.

Automation of Declare Model Construction

In the current version of the approach, humans have to manually select the association and rules and sequential patterns they want to transform to Declare model elements. For patterns with support lower than one, the confidence value has to be determined to assess their significance for process descriptions. Such selections require some effort, domain knowledge, and expertise in translating the rules and patterns to Declare model elements. Since at least initially, most likely process analysts are

responsible for such activities, it may be an acceptable procedure. However, even for them an automatic transformation approach that constructs a first version of the declarative model that they can adapt afterwards would be convenient. This is even more true when employees with no or only limited data mining or process analysis background apply the techniques on event logs.

Further Integration in BPM Activities

Section 8.2 has provided a first impression of how an integration of declarative process mining into current BPM structures could look like. Fahland et al. [FLM+09, FMR+09] address the issues of understandability and maintainability for the imperative and declarative paradigm and touch psychological and social work theory aspects. Andaloussi et al. [ADB+20] go in a similar direction and investigate the quality of declarative models. The result is a framework including several dimensions for quality assessment.

Although these contributions cover essential aspects of the relationship between the imperative and declarative paradigm, they do not focus on implications for practical use. More research should be conducted to develop guidelines for combining declarative modeling with the existing BPM activities of a company through, for instance, hybrid process specifications. For this purpose, case studies about the application of such integration should be analyzed to develop a general framework or similar concept. In sum, an inductive research approach with trial-and-error in practice and theory formation afterwards may be most promising here.

Quantifying the Benefit of Declarative Process Mining in Practice

Another aspect related to the integration into the BPM structure is a quantification of the benefit it brings. Chapter 1 has raised whether applying association rule and sequential pattern mining is beneficial and for whom.

Besides process performance (e.g., lead time, error rate) and long-term costs aspects, there are more direct measurements to evaluate the benefit of declarative mining techniques. One could be the average time of an analysis project for a new process. An alternative idea is to establish proxies that measure satisfaction with declarative techniques. Surveys with process analysts and employees with no BPM expertise could deliver quantitative estimations about the benefits of declarative process mining in practice.

All in all, applying fundamental data mining techniques like association rule and sequential pattern mining to event logs in the way described in this thesis may have promising benefits for BPM and the digitalization progress of business processes. In the future, case studies that apply the approach to real-world event logs could deliver more substantial insights regarding its use and advantages in practice.

Bibliography

[ABMvdA20] Anti Alman, Karl Johannes Balder, Fabrizio Maria Maggi, and Han van der Aa. Declo: A chatbot for user-friendly specification of declarative process models. In *Dissertation Award, Doctoral Consortium, and Demonstration Track at BPM 2020 co-located with 18th International Conference on Business Process Management*, 2020.

[ADB+20] Amine Abbad Andaloussi, Christopher J Davis, Andrea Burattin, Hugo A López, Tijs Slaats, and Barbara Weber. Understanding quality in declarative process modeling through the mental models of experts. In *International Conference on Business Process Management*, pages 417–434. Springer, 2020.

[ADCH+] Anti Alman, Claudio Di Ciccio, Dominik Haas, Fabrizio Maria Maggi, and Alexander Nolte. Rule mining with RuM. In *2nd International Conference on Process Mining, 2020*.

[ADCM+21] Anti Alman, Claudio Di Ciccio, Fabrizio Maria Maggi, Marco Montali, and Han van der Aa. RuM: Declarative process mining, distilled. In Artem Polyvyanyy, Moe Thandar Wynn, Amy Van Looy, and Manfred Reichert, editors, *Business Process Management*, pages 23–29. Springer International Publishing, 2021.

[AIS93] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

[AS+94] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proceedings of the 20th Conference on Very Large Data Bases, VLDB*, volume 1215, pages 487–499, 1994.

[AS95] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14. IEEE, 1995.

[ASA+19] Zaher Ali Al-Sai, Rosni Abdullah, et al. A review on big data maturity models. In *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, pages 156–161. IEEE, 2019.

[BGvdW09] Melike Bozkaya, Joost Gabriels, and Jan Martijn van der Werf. Process diagnostics: a method based on process mining. In *2009 International Conference on Information, Process, and Knowledge Management*, pages 22–27. IEEE, 2009.

[BKMZ15] David Basin, Felix Klaedtke, Samuel Müller, and Eugen Zălinescu. Monitoring metric first-order temporal properties. *Journal of the ACM*, 62(2), 2015.

[BMS16] Andrea Burattin, Fabrizio M. Maggi, and Alessandro Sperduti. Conformance checking based on multi-

perspective declarative process models. *Expert Systems with Applications*, 65:194–211, 2016.

[BMUT97] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. SIGMOD '97, page 255–264. Association for Computing Machinery, 1997.

[CBM18] Luca Canetta, Andrea Barni, and Elias Montini. Development of a digitalization maturity model for the manufacturing sector. In *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 1–7. IEEE, 2018.

[CWWM14] Anne Katharina Cleven, Robert Winter, Felix Wortmann, and Tobias Mettler. Process management in hospitals: an empirically grounded maturity model. *Business Research*, 7(2):191–216, 2014.

[DCM13] Claudio Di Ciccio and Massimo Mecella. A two-step fast algorithm for the automated discovery of declarative workflows. In *2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 135–142. IEEE, 2013.

[DCMNT17] Anna De Carolis, Marco Macchi, Elisa Negri, and Sergio Terzi. A maturity model for assessing the digital readiness of manufacturing companies. In *IFIP International Conference on Advances in Production Management Systems*, pages 13–20. Springer, 2017.

[DE00] Jörg Desel and Thomas Erwin. Hybrid specifications: looking at workflows from a run-time perspective. *Com-*

puter Systems Science and Engineering, 15(5):291–302, 2000.

[DGDMM15] Giuseppe De Giacomo, Marlon Dumas, Fabrizio Maria Maggi, and Marco Montali. Declarative process modeling in bpmn. In Jelena Zdravkovic, Marite Kirikova, and Paul Johannesson, editors, *Advanced Information Systems Engineering*, pages 84–100. Springer International Publishing, 2015.

[DHLU17] Søren Debois, Thomas T Hildebrandt, Paw Høvsgaard Laursen, and Kenneth Ry Ulrik. Declarative process mining for dcr graphs. In *Proceedings of the Symposium on Applied Computing*, pages 759–764, 2017.

[DLRM$^+$13] Marlon Dumas, Marcello La Rosa, Jan Mendling, Hajo A Reijers, et al. *Fundamentals of Business Process Management*. Springer, 2013.

[DMMM14] Riccardo De Masellis, Fabrizio M Maggi, and Marco Montali. Monitoring data-aware business constraints with finite state automata. In *Proceedings of the 2014 International Conference on Software and System Process*, pages 134–143, 2014.

[dMRVDA19] Eduardo González López de Murillas, Hajo A Reijers, and Wil MP Van Der Aalst. Connecting databases with process mining: a meta model and toolset. *Software & Systems Modeling*, 18(2):1209–1247, 2019.

[DS15] Søren Debois and Tijs Slaats. The analysis of a real life declarative process. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 1374–1382. IEEE, 2015.

[DSL+19] Dusanka Dakic, Darko Stefanovic, Teodora Lolic, Dajana Narandzic, and Nenad Simeunovic. Event log extraction for the purpose of process mining: a systematic literature review. In *International Symposium in Management Innovation for Sustainable Management and Entrepreneurship*, pages 299–312. Springer, 2019.

[DVdATH05] Marlon Dumas, Wil M Van der Aalst, and Arthur H Ter Hofstede. *Process-aware information systems: bridging people and software through process technology*. John Wiley & Sons, 2005.

[FAS19] Vanessa Felch, Björn Asdecker, and Eric Sucky. Maturity models in the age of industry 4.0–do the available models correspond to the needs of business practice? In *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

[FG19] Valeria Fionda and Antonella Guzzo. Control-flow modeling with Declare: Behavioral properties, computational complexity, and tools. *IEEE Transactions on Knowledge and Data Engineering*, 32(5):898–911, 2019.

[FHW16] Eibe Frank, Mark Hall, and Ian Witten. *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*. Morgan Kaufmann, 2016.

[FLM+09] Dirk Fahland, Daniel Lübke, Jan Mendling, Hajo Reijers, Barbara Weber, Matthias Weidlich, and Stefan Zugal. Declarative versus imperative process modeling languages: The issue of understandability. In *Enterprise,*

Business-Process and Information Systems Modeling, pages 353–366. Springer, 2009.

[FMR+09] Dirk Fahland, Jan Mendling, Hajo A Reijers, Barbara Weber, Matthias Weidlich, and Stefan Zugal. Declarative versus imperative process modeling languages: The issue of maintainability. In *International Conference on Business Process Management*, pages 477–488. Springer, 2009.

[FVGZT14] Philippe Fournier-Viger, Ted Gueniche, Souleymane Zida, and Vincent S Tseng. Erminer: sequential rule mining using equivalence classes. In *International Symposium on Intelligent Data Analysis*, pages 108–119. Springer, 2014.

[FVLR+17] Philippe Fournier Viger, Chun-Wei Lin, Uday Rage, Yun Sing Koh, and Rincy Thomas. A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1:54–77, 2017.

[FVNT11] Philippe Fournier-Viger, Roger Nkambou, and Vincent Shin-Mu Tseng. Rulegrowth: mining sequential rules common to several sequences by pattern-growth. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 956–961, 2011.

[GPC99] Orna Grumberg, Doron A Peled, and Edmund Clarke. *Model checking*. MIT Press, 1999.

[Gro16] XES Working Group. IEEE standard for extensible event stream (XES) for achieving interoperability in event logs and event streams. *IEEE Std 1849-2016*, pages 1–50, 2016.

[Ham90] Michael Hammer. Reengineering work: don't automate, obliterate. *Harvard business review*, 68(4):104–112, 1990.

[HM11] Thomas T Hildebrandt and Raghava Rao Mukkamala. Declarative event-based workflow as distributed dynamic condition response graphs. *arXiv preprint arXiv:1110.4161*, 2011.

[HPY00] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *ACM sigmod record*, 29(2):1–12, 2000.

[HPYM04] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery*, 8(1):53–87, 2004.

[KMDCDF16] Taavi Kala, Fabrizio Maria Maggi, Claudio Di Ciccio, and Chiara Di Francescomarino. Apriori and sequence analysis for discovering declarative process models. In *2016 IEEE 20th International Enterprise Distributed Object Computing Conference (EDOC)*, pages 1–9. IEEE, 2016.

[Koy90] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[KPP03] Walter A Kosters, Wim Pijls, and Viara Popova. Complexity analysis of depth first and FP-Growth implementations of Apriori. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 284–292. Springer, 2003.

[LFvdA13] Sander JJ Leemans, Dirk Fahland, and Wil MP van der Aalst. Discovering block-structured process models from event logs-a constructive approach. In *International Conference on Applications and Theory of Petri Nets and Concurrency*, pages 311–329. Springer, 2013.

[LKW09] David Lo, Siau-Cheng Khoo, and Limsoon Wong. Non-redundant sequential rules—theory and algorithm. *Information Systems*, 34(4-5):438–453, 2009.

[LRU20] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 3rd edition, 2020.

[Mag13] Fabrizio Maria Maggi. Declarative process mining with the declare component of ProM. *International Conference on Business Process Management (Demos), BPM*, 2013.

[MBvdA12] Fabrizio M Maggi, RP Jagadeesh Chandra Bose, and Wil MP van der Aalst. Efficient discovery of understandable declarative process models from event logs. In *International Conference on Advanced Information Systems Engineering*, pages 270–285. Springer, 2012.

[MDGBM13] Fabrizio Maria Maggi, Marlon Dumas, Luciano García-Bañuelos, and Marco Montali. Discovering data-aware declarative process models from event logs. In *Business Process Management*, pages 81–96. Springer, 2013.

[MMB19] Fabrizio Maria Maggi, Marco Montali, and Ubaier Bhat. Compliance monitoring of multi-perspective declarative process models. In *2019 IEEE 23rd International Enterprise*

Distributed Object Computing Conference (EDOC), pages 151–160. IEEE, 2019.

[MMvdA11] Fabrizio M Maggi, Arjan J Mooij, and Wil MP van der Aalst. User-guided discovery of declarative process models. In *2011 IEEE symposium on computational intelligence and data mining (CIDM)*, pages 192–199. IEEE, 2011.

[MSW11] Andreas Meyer, Sergey Smirnov, and Mathias Weske. Data in business processes. *EMISA Forum*, 31:5–31, 2011.

[MTIV97] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[NCB+20] Nicolò Navarin, Matteo Cambiaso, Andrea Burattin, Fabrizio M Maggi, Luca Oneto, and Alessandro Sperduti. Towards online discovery of data-aware declarative process models from event streams. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[Ost04] Alexander Osterwalder. *The business model ontology a proposition in a design science approach*. PhD thesis, University of Lausanne, Switzerland, 2004.

[PB18] Diogo Proença and José Borbinha. Maturity models for data and information management. In *International Conference on Theory and Practice of Digital Libraries*, pages 81–93. Springer, 2018.

[PCWP+11] Ricardo Pérez-Castillo, B. Weber, J. Pinggera, S. Zugal, I. G. D. Guzmán, and M. Piattini. Generating event logs

from non-process-aware systems enabling business process mining. *Enterprise Information Systems*, 5:301 – 335, 2011.

[PDCM14] Johannes Prescher, Claudio Di Ciccio, and Jan Mendling. From declarative processes to imperative models. *Fourth International Symposium on Data-driven Process Discovery and Analysis (SIMPDA)*, 1293:162–173, 2014.

[Pet62] Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Technische Hochschule Darmstadt, 1962.

[PHMA+01] Jian Pei, Jiawei Han, B. Mortazavi-Asl, H. Pinto, Qiming Chen, U. Dayal, and Mei-Chun Hsu. Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings 17th International Conference on Data Engineering*, pages 215–224, 2001.

[Pia91] Gregory Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, pages 229–248. AAAI/MIT Press, 1991.

[PLHV14] Thi-Thiet Pham, Jiawei Luo, Tzung-Pei Hong, and Bay Vo. An efficient method for mining non-redundant sequential rules using attributed prefix-trees. *Engineering Applications of Artificial Intelligence*, 32:88–99, 2014.

[PSVdA07] Maja Pesic, Helen Schonenberg, and Wil MP Van der Aalst. Declare: Full support for loosely-structured processes. In *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC)*, pages 287–287. IEEE, 2007.

[PWZ+11] Paul Pichler, Barbara Weber, Stefan Zugal, Jakob Pinggera, Jan Mendling, and Hajo A Reijers. Imperative versus declarative process modeling languages: An empirical investigation. In *International Conference on Business Process Management*, pages 383–394. Springer, 2011.

[Rei20] Lars Reinkemeyer. *Process Mining in Action: Principles, Use Cases and Outlook*. Springer Nature, 2020.

[RF12] Álvaro Rebuge and Diogo R Ferreira. Business process analysis in healthcare environments: A methodology based on process mining. *Information Systems*, 37(2):99–116, 2012.

[RMDLVDA15] Marcella Rovani, Fabrizio M Maggi, Massimiliano De Leoni, and Wil MP Van Der Aalst. Declarative process mining in healthcare. *Expert Systems with Applications*, 42(23):9236–9251, 2015.

[RMGSC16] Eric Rojas, Jorge Munoz-Gama, Marcos Sepúlveda, and Daniel Capurro. Process mining in healthcare: A literature review. *Journal of biomedical informatics*, 61:224–236, 2016.

[RVdA08] Anne Rozinat and Wil MP Van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008.

[SA96] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *International Conference on Extending Database Technology*, pages 1–17. Springer, 1996.

[SCJM15] Stefan Schönig, Cristina Cabanillas, Stefan Jablonski, and Jan Mendling. Mining the organisational perspective in agile business processes. In *Enterprise, Business-Process and Information Systems Modeling*, pages 37–52. Springer, 2015.

[SDCMM16] Stefan Schönig, Claudio Di Ciccio, Fabrizio M Maggi, and Jan Mendling. Discovery of multi-perspective declarative process models. In *International Conference on Service-Oriented Computing*, pages 87–103. Springer, 2016.

[SM17] Anand Swaminathan and Jürgen Meffert. *Digital@ Scale: the playbook you need to transform your company*. John Wiley & Sons, 2017.

[SRSC+16] Stefan Schönig, Andreas Rogge-Solti, Cristina Cabanillas, Stefan Jablonski, and Jan Mendling. Efficient and customisable declarative process mining with sql. In *International Conference on Advanced Information Systems Engineering*, pages 290–305. Springer, 2016.

[SSMR16] Tijs Slaats, Dennis MM Schunselaar, Fabrizio M Maggi, and Hajo A Reijers. The semantics of hybrid process models. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, pages 531–551. Springer, 2016.

[SVOK12] Frank Schönthaler, Gottfried Vossen, Andreas Oberweis, and Thomas Karle. *Business Processes for Business Communities*. Springer, 2012.

[TMB20] Tristan Thordsen, Matthias Murawski, and Markus Bick. How to measure digitalization? a critical evaluation of

digital maturity models. *Responsible Design, Implementation and Use of Information and Communication Technology*, 12066:358, 2020.

[TSK16] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.

[VBVDVDA10] HMW Verbeek, Joos CAM Buijs, Boudewijn F Van Dongen, and Wil MP Van Der Aalst. XES, XESame, and ProM 6. In *International Conference on Advanced Information Systems Engineering*, pages 60–75. Springer, 2010.

[vdAAdM+12] Wil van der Aalst, Arya Adriansyah, Ana Karla Alves de Medeiros, Franco Arcieri, et al. Process Mining Manifesto. In *Business Process Management Workshops*, volume 99, pages 169–194. Springer Berlin Heidelberg, 2012.

[vdABMN20] Han van der Aa, Karl Johannes Balder, Fabrizio Maria Maggi, and Alexander Nolte. Say it in your own words: Defining declarative process models using speech recognition. In Dirk Fahland, Chiara Ghidini, Jörg Becker, and Marlon Dumas, editors, *Business Process Management Forum*, pages 51–67, Cham, 2020. Springer International Publishing.

[vdABvZ17] Wil M. P. van der Aalst, Alfredo Bolt, and Sebastiaan J. van Zelst. RapidProM: Mine your processes and not just your data. arXiv:1703.03740v1, 2017.

[vdADCLR19] Han van der Aa, Claudio Di Ciccio, Henrik Leopold, and Hajo A Reijers. Extracting declarative process models

from natural language. In *International Conference on Advanced Information Systems Engineering*, pages 365–382. Springer, 2019.

[vDAPS09] Wil MP van Der Aalst, Maja Pesic, and Helen Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science-Research and Development*, 23(2):99–113, 2009.

[VdAVDH+03] Wil MP Van der Aalst, Boudewijn F Van Dongen, Joachim Herbst, Laura Maruster, Guido Schimm, and Anton JMM Weijters. Workflow mining: A survey of issues and approaches. *Data & Knowledge Engineering*, 47(2):237–267, 2003.

[VdAWM04] Wil 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.

[vDS15] Boudewijn F van Dongen and Shiva Shabani. Relational XES: Data management for process mining. In *CAiSE Forum*, volume 2015, pages 169–176, 2015.

[VEGD20] Gottfried Vossen, Jan Everding, Nico Grohmann, and Stuart Dillon. Assessing Levels of Digital Maturity Through a Digitalization Check. *ResearchGate*, 2020.

[VELLVDA15] Maikel L Van Eck, Xixi Lu, Sander JJ Leemans, and Wil MP Van Der Aalst. PM^2: a process mining project methodology. In *International Conference on Advanced Information Systems Engineering*, pages 297–313. Springer, 2015.

[VL10] Amy Van Looy. Does it matter for business process maturity? a comparative study on business process maturity models. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, pages 687–697. Springer, 2010.

[VL19] Gottfried Vossen and Jens Lechtenbörger. Structuring what you are doing: 20 years of business process modelling. In *The Art of Structuring*, pages 227–238. Springer, 2019.

[Wil16] Wil van der Aalst. *Process Mining - Data Science in Action.* Springer-Verlag Berlin Heidelberg, 2016.

[WM12] Michael Westergaard and Fabrizio Maria Maggi. Looking into the future: using timed automata to provide a priori advice about timed declarative process models. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, pages 250–267. Springer, 2012.

[WR11] AJMM Weijters and Joel Tiago S Ribeiro. Flexible heuristics miner (FHM). In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 310–317. IEEE, 2011.

[WSR13] Michael Westergaard, Christian Stahl, and Hajo A Reijers. Unconstrainedminer: efficient discovery of generalized declarative process models. *BPM Center Report, No. BPM-13-28*, 207, 2013.

[Zak01] Mohammed J Zaki. Spade: An efficient algorithm for

mining frequent sequences. *Machine Learning*, 42(1):31–60, 2001.

[ZB03] Qiankun Zhao and Sourav S Bhowmick. Association rule mining: A survey. *Nanyang Technological University, Singapore*, 2003.

[ZSJ14] Michael Zeising, Stefan Schönig, and Stefan Jablonski. Towards a common platform for the support of routine and agile business processes. In *10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 94–103. IEEE, 2014.

List of Web Pages

[1] Clayton Christensen: Disruptive Innovation. http://claytonchristensen.com/key-concepts/. Version: 2021. - Last accessed: 03-12-2021.

[2] Object Management Group: About the Business Process Model and Notation Specification Version 2.0.2. https://www.omg.org/spec/BPMN/. Version: 2021. - Last accessed: 03-19-2021.

[3] Lucidchart: Was ist Business Process Modeling Notation (BPMN)? https://www.lucidchart.com/pages/de/was-ist-business-process-modeling-notation. Version: 2021. - Last accessed: 03-19-2021.

[4] Fluxicon: Disco. https://fluxicon.com/disco/. Version: 2021. - Last accessed: 03-16-2021.

[5] Celonis SE: Celonis. https://www.celonis.com/de. Version: 2021. - Last accessed: 03-16-2021.

[6] RapidMiner, Inc.: RapidMiner. https://rapidminer.com/. Version: 2021. - Last accessed: 04-07-2021.

[7] ProM: The Process Mining Toolkit (ProM). https://www.promtools.org/doku.php. Version: 2021. - Last accessed: 04-28-2021.

[8] 4TU.ResearchData: Third International Business Process Intelligence Challenge. https://data.4tu.nl/articles/dataset/BPI_

Challenge_2013_incidents/12693914/1. Version 2013. - Last accessed: 05-25-2021.

[9] Eindhoven University of Technology: Event logs and models used in Process Mining book. http://www.processmining.org/event_logs_and_models_used_in_book. Version: 2016. - Last accessed: 05-25-2021.

[10] Horus Software GmbH: Business Modeler. https://www.horus.biz/en/products/. 2021. - Last accessed: 07-14-2021.

[11] HSPI S.p.A.: Process Mining: A Database of Applications. https://www.hspi.it/wp-content/uploads/2020/01/HSPI_Process_Mining_Database2020.pdf. Version: 2020. - Last accessed: 08-02-2021.

[12] Marcos Sponton: Machine Learning ROI: How to model the investment value of Machine Learning? https://blog.usejournal.com/machine-learning-roi-how-to-model-the-investment-value-of-machine-learning-a29be1f2f827. Version: 2018. - Last accessed: 08-10-2021.

[13] Wikipedia: Apriori algorithm. https://en.wikipedia.org/wiki/Apriori_algorithm. Version: 2021. - Last accessed: 09-13-2021.

[14] Prof. Dr. Christian Bizer: Association Analysis. https://www.uni-mannheim.de/media/Einrichtungen/dws/Files_Teaching/Data_Mining/FSS2019/DM07-Association-Analysis-FSS2019.pdf. Version: 2019. - Last accessed: 09-13-2021.

[15] Saul Dobilas: Apriori Algorithm for Association Rule Learning — How To Find Clear Links Between Transactions. https://towardsdatascience.com/apriori-algorithm-for-

association-rule-learning-how-to-find-clear-links-between-transactions-bf7ebc22cf0a. Version: 2021. - Last accessed: 09-14-2021.

[16] Philippe Fournier-Viger: An Introduction to Sequential Rule Mining. https://data-mining.philippe-fournier-viger.com/introduction-to-sequential-rule-mining/. Version: 2015. - Last accessed: 10-07-2021.

[17] Philippe Fournier-Viger: SPMF - An Open-Source Data Mining Library. https://www.philippe-fournier-viger.com/spmf/. Version: 2021. - Last accessed: 10-07-2021.

[18] 4TU.ResearchData: BPI Challenge 2019. https://data.4tu.nl/articles/dataset/BPI_Challenge_2019/12715853/1. Version: 2019. - Last accessed: 11-11-2021.

[19] McKinsey & Company: Digital 20/20. https://www.mckinsey.com/business-functions/mckinsey-digital/how-we-help-clients/digital-2020/our-assessments/strategy. Version: 2021. - Last accessed: 11-17-2021.

[20] 4TU.ResearchData: Production Analysis with Process Mining Technology. https://data.4tu.nl/articles/dataset/Production_Analysis_with_Process_Mining_Technology/12697997. Version 2014. - Last accessed: 11-22-2021.

[21] Sebastiaan van Zelst; Alfredo BoltLast: Bringing Process Mining to Analytic Workflows. http://www.rapidprom.org/. Version: 2021. - Last accessed: 11-22-2021.

[22] KNIME AG: End to End Data Science. https://www.knime.com/. Version: 2021. - Last accessed: 11-29-2021.

[23] SAS Institute Inc.: Analytics Software & Solutions. https://www.sas.com/. Version: 2021. - Last accessed: 11-29-2021.

[24] Brightpearl, Inc. The Ultimate Guide to Order to Cash (O2C) Process. https://www.brightpearl.com/blog/the-ultimate-guide-to-order-to-cash-o2c-process. Version: 2021. - Last accessed: 11-29-2021.

[25] Gartner, Inc.: Gartner Glossary - Digitalization. https://www.gartner.com/en/information-technology/glossary/digitalization. Version: 2021. - Last accessed: 12-02-2021.

[26] Compact: Process Mining - Let data describe your process. https://www.compact.nl/articles/process-mining/. Version: 2021. - Last accessed: 12-03-2021.

List of Abbreviations

BI	Business Intelligence.
BPM	Business Process Management.
BPMN	Business Process Model and Notation.
CoE	Center of Excellence.
DCR	Dynamic Condition Response.
EDI	Electronic Data Interchange.
EPC	Event-driven Process Chain.
ETL	Extract, Transform, Load.
IT	Information Technology.
KPI	Key Performance Indicator.
LTL	Linear Temporal Logic.
MFOTL	Metric First Order Temporal Logic.
NLP	Natural Language Processing.
PAIS	Process-aware Information System.

SMEs Small and Medium-sized Enterprises.
SQL Structured Query Language.

XES eXtensible Event Stream.

A Overview of the Declare Template Set

Table A.1: Unary Relations in the Declare Template Set (based on [DCM13, KMDCDF16, FG19]).

Template	Definition
EXISTENCE (A)	A has to occur at least once.
EXISTENCE2 (A)	A has to occur at least twice.
ABSENCE (A)	A can never happen.
ABSENCE2 (A)	A can happen at most once.
INIT (A)	A process has to start with activity A.
END (A)	A process has to end with activity A.

Table A.2: Binary Relations in the Declare Template Set (based on [DCM13, KMDCDF16, FG19]).

Template	Definition
CHOICE (A,B)	A or B or both must occur.
EXCLUSIVE CHOICE (A,B)	Either A or B must occur.
CO-EXISTENCE (A,B)	If A or B occurs, the respective other one must occur.
PRECEDENCE (A,B)	B can only occur if A has occurred before.
SUCCESSION (A,B)	If A occurs, B must eventually follow and B cannot occur if A has not occurred before.
RESPONSE (A,B)	If A occurs, B must eventually follow, without any other A in between.
ALTERNATE RESPONSE (A,B)	If A occurs, B must eventually follow, without any other A in between.
ALTERNATE PRECEDENCE (A,B)	B can occur only if A has occurred before, without any other B in between.
ALTERNATE SUCCESSION (A,B)	If A occurs, B must eventually follow, without any other A in between and B can occur only if A has occurred before, without any other B in between.
CHAIN RESPONSE (A,B)	If A occurs, B must occur next.

Table A.3: Binary Relations in the Declare Template Set (based on [DCM13, KMDCDF16, FG19]), continued.

Template	Definition
CHAIN PRECEDENCE (A,B)	B can only occur immediately after A.
CHAIN SUCCESSION (A,B)	If A occurs, B must occur next and B can only occur immediately after A.
NOT CO-EXISTENCE (A,B)	If A occurs, B cannot occur and the other way around.
NOT SUCCESSION (A,B)	If A occurs, B cannot eventually follow and A cannot occur before B.
NOT CHAIN SUCCESSION (A,B)	If A occurs, B cannot occur next and A cannot occur immediately before B.

Fundamental Data Mining Techniques for Declarative Process Mining

Nico Grohmann

Process mining is a Business Process Management (BPM) technique that uses execution data of business processes for their analysis. By transforming the data to so-called event logs, process mining tools generate process models that describe the executions as close as possible. Process discovery can result either in graph-based notations (e.g., Petri nets or BPMN) or declarative ones like Declare. One hypothesis in this work is that declarative constraint templates can support model understanding in case process mining results in large, confusing “spaghetti” diagrams. Overall, this work contributes an approach including a prototypical implementation for applying association rule and sequential pattern mining to event logs for discovering declarative process models. Preprocessing steps and the transformation of rules and patterns to constraints in Declare are addressed explicitly. In this way, analysts receive transparent insights into the basis of the overall declarative model.

ISBN 978-3-8405-0266-8