MASTER

Conformance checking with incongruous input
a systematic approach for uncovering actual process nonconformities in practice

van der Laan, D.

Award date:
2019

Conformance Checking with Incongruous Input

A Systematic Approach for Uncovering Actual Process Nonconformities in Practice

D. (Daniël) van der Laan

in partial fulfillment of the requirements for the degree of

Master of Science
in
Business Information Systems

Supervisors:
prof.dr.ir. B.F. (Boudewijn) van Dongen (TU/e)
dr.ir. H. (Rik) Eshuis (TU/e)
J.G.P. (Joris) van de Veerdonk MSc (Protiviti)

Version 1.1

Eindhoven, January 2019

Abstract

Business processes need to be continuously monitored and assessed to ensure operational success. Ensuring that processes are executed correctly can be done by analyzing whether processes adhere to their original design. Conformance checking is a subfield of process mining that allows for automatic detection of process nonconformity. Traditionally, two input components are needed to apply conformance checking techniques: a process model that describes the intended process flow, and an event log with empirical process data. However, process models and event logs are not always fully reliable in practice. Although they describe the same process, an event log and a process model often do not match well. This work has described such a log-model pair as *incongruous.* Current conformance checking techniques are incapable of providing meaningful insights with incongruous input.

In this research, a systematic approach was developed that allows use of present-day conformance checking techniques on incongruous input after adaptation of the input. The approach was developed on a self-created example. The effectiveness of the approach was determined by applying the approach in a case study on a purchase-to-pay process in collaboration with multinational food company Wessanen. An incongruous log-model pair was used as input, with an event log created from empirical ERP data and the process models made available by Wessanen. By leveraging a third input, namely *domain knowledge*, meaningful results came out of the conformance analysis. This domain knowledge was gained in collaboration with both global consulting firm Protiviti and Wessanen.

The systematic approach developed is an iterative process in which congruity issues are identified by means of log replay techniques. During the identification phase, appearing nonconformities are classified in one of three ways: resolvable incongruity (to resolve by adapting log/model), irresolvable incongruity (to whitelist), or nonconformity (to blacklist). By subsequently adapting the event log and process model to eliminate resolvable incongruities, it is possible to create a new log-model pair that became more congruous every iteration. Blacklisted nonconformities are real process nonconformities that should not occur, and these have to be continuously be retrieved as such. The remaining (irresolvable) incongruities can be whitelisted if the used tooling allows this. In this work, process mining tool Celonis was used to account for this.

This work has built in the direction of generalized conformance checking, in which conformance checking techniques can be applied on unreliable input in practice. Results indicated that the approach can be perceived as useful, and that it can be used on different processes. This thesis is concluded by discussing some interesting limitations and possible future research outlooks, to further progress the field of conformance checking.

Preface

This work marks the end of my period as a student at the Eindhoven University of Technology. During this time, I have studied to receive my Bachelor's degree in Computer Science and Engineering, after which I enrolled in a master's program at the TU/e. During my master's program I was fortunate enough to study abroad at the Queensland University of Technology as part of my degree. Now, I am about to earn my Master's degree in Business Information Systems. For the past months, I have been working in collaboration with Protiviti B.V., Royal Wessanen nv, Celonis B.V., and the Analytics for Information Systems group of the university to conduct an interesting research project. The thesis that lies in front of you is the result of this labor, and it would not have been in this shape without the help of others. Therefore, I would like to take this opportunity to thank the people that have helped me get to this point.

First, I would like to thank Boudewijn van Dongen, who acted as my graduation supervisor during the project. His expertise in the field of conformance checking has helped make this project extremely interesting with a result that is academically relevant. His guidance and feedback has been crucial to make this thesis the way it is.

I would also like to thank Ernst Stoelhorst for convincing me to undertake this graduation project as part of Protiviti. Everyone at Protiviti has taken me into the family from the first day and I could not have wished for a more supportive environment for my graduation project. Regarding the day-to-day of the project itself, I would like to give special thanks to Joris van de Veerdonk. Joris was closely involved with my project from the start, and I have to thank him for providing me with guidance along the way, mentoring me, and making me become better suited for a professional career after this.

Furthermore, my thanks go out to Christiaan Koster for being incredibly open to collaborate with Protiviti and me on this project. Christiaan has taken time out of his busy schedule to help me receive the help I needed from all the people involved at Wessanen, giving me all the business and technical knowledge I needed. Also a big thanks to Susanne McGregor-Stevens for being a big part of the many discussions we had about the process mining results.

Then, thanks to Janina Nakladal and Simon Riezebos from Celonis for allowing me to use market-leading software to conduct this research, and for providing me with the help needed when technical problems stood in my way.

Finally, I want to thank my family for always supporting me in all imaginable ways, and all my friends for helping me unwind when I feel I am taking life a little too seriously. I am both happy and proud to have reached this point in my life, and I am grateful for all those who helped me get here. I enjoyed this chapter of my life tremendously, and I am excited for the challenges that lie ahead.

Daniël van der Laan
Eindhoven, 2018

Contents

Contents **vii**

List of Figures **ix**

1 Introduction **1**
1.1 Introduction . . . 1
1.2 Thesis outline . . . 2
1.3 Protiviti . . . 3
1.4 Wessanen . . . 3
1.5 Scope . . . 3
1.6 Problem statement . . . 5
1.7 Research Objectives . . . 6
1.7.1 Academic relevance . . . 6
1.7.2 Industrial relevance . . . 6
1.8 Methodology . . . 6
1.9 Running example . . . 7

2 Theoretical Background **11**
2.1 Conformance, compliance, and control . . . 11
2.1.1 Definitions . . . 11
2.1.2 Conformance and Compliance . . . 12
2.1.3 Constraint dimensions . . . 13
2.1.4 Internal Control . . . 14
2.1.5 Questions to answer . . . 15
2.2 Process Mining . . . 16
2.2.1 Overview . . . 16
2.2.2 Event log . . . 17
2.3 Conformance checking . . . 18
2.3.1 Overview . . . 18
2.3.2 Conformance metrics . . . 19
2.3.3 Techniques . . . 19
2.4 Generalized conformance checking . . . 21

3 Incongruity: Causes and Context **23**
3.1 Cause of Incongruity . . . 23
3.1.1 Event log quality . . . 25
3.1.2 Process model quality . . . 26
3.2 Possible Congruity Issues . . . 28
3.2.1 Too granular/too abstract . . . 29
3.2.2 Too many/too few process steps . . . 29
3.2.3 Syntactic issues . . . 30
3.2.4 Semantic issues . . . 30

3.3 Domain Knowledge 31

4 Identifying Log-Model Pair Congruity Issues 33
4.1 Overview 33
4.2 Pattern Recognition 34
4.3 Issue Identification and Classification 36

5 Adaptation Process 39
5.1 Systematic Approach 39
5.2 Adapting the Log and Model 41
5.2.1 Event log 41
5.2.2 Process model 42
5.2.3 Approaches 43
5.3 Step-by-step Approach 44

6 Case Study 45
6.1 Business and Data Understanding 45
6.2 Data Preparation 45
6.2.1 Event log 45
6.2.2 Process Model 46
6.3 Congruity Issue Identification 47
6.4 Adapting the Model-Log Pair 49
6.5 Results 50

7 Conclusion 53
7.1 Conclusion 53
7.2 Contribution 53
7.3 Limitations and Future Work 54

Bibliography 57

Appendix 59

A Running Example 59

B Event Log Maturity Model 64

C Case study 66

List of Figures

1.1 Research scope in the context of the BPM life-cycle model (Dumas et al., 2013) . . 4
1.2 Cross-Industry Standard Process for Data Mining (CRISP-DM) 7
1.3 Excerpt of process model used as running example 9

2.1 Compliance Management life cycle ((Ramezani Taghiabadi, 2017)) 12
2.2 Process mining overview (visual adaptation from the Process Mining Manifesto) . 16
2.3 Conformance checking input and output 18

3.1 Business process to event log class diagram (adapted version from Process Mining book) 23
3.2 Different formality levels in process models. a) Directly-follows graph. b) EPC diagram (operational model). c) BPMN model (executable model). 26
3.3 Discrepancies relative to the ideal component 28
3.4 Third input for generalized conformance checking: domain knowledge 31

4.1 Alignment of a case: nonconformity on log event *Scan ticket* highlighted 33
4.2 Three alignments of the running example 34
4.3 Pattern identified at start of running example 35
4.4 Second pattern identified in the running example 35
4.5 Starting activities in the model 36
4.6 Classification decision tree 37
4.7 Alignment of case 10 - Log move on *Cancel approval* highlighted 37
4.8 Model excerpt with request approval and compensation payment 38
4.9 Alignment of case 10 - Sync move on *Pay compensation* highlighted 38

5.1 Iterative approach to conformance checking on incongruous input 39
5.2 SQL query for adaptation of the event log for a semantic issue 42
5.3 SQL query for adaptation of the event log for a syntactic issue 42
5.4 Adaptation of the event log for an abstraction issue: grouped activity added 43

6.1 Excerpt of P2P event log created from Wessanen ERP data 46
6.2 Excerpt of P2P EPC model made available by Wessanen 47
6.3 Initial alignment input log and model, case 500450022286600030 47
6.4 Semantic nonconformity identification, case 500450022286600030 48
6.5 Syntactic nonconformity identification, case 500450022155500010 48
6.6 Model adaptation required 49
6.7 Initial results of adaptation, alignment on case 500450022286600030 50
6.8 Initial conformance results in Celonis 51
6.9 Conformance results in Celonis after applying systematic approach 51
6.10 Generated insights by Celonis 51

A.1 Starting point for the running example process model 60
A.2 Running example BPMN model 61

A.3 Running example reduced Petri net 62
A.4 Running example reduced Petri net with condensed formatting 63

C.1 Process model discovered from Wessanen ERP data 67
C.2 Unadapted process model as provided by Wessanen 68
C.3 Translated, unadapted process model 69
C.4 Translated, unadapted process model as Petri net 70

Chapter 1

Introduction

In many organizations, effective processes are at the core of operational success (Kirchmer, 2011). This could be processes of various kinds, such as producing products, supply chain, purchasing or selling products, providing services, or customer support. Therefore, it is important for organizations to carefully design and implement their processes. However, it does not end there. The effectiveness and efficiency of processes should be continuously monitored, analyzed, and improved in order to maintain and grow business Accorsi (2011). Business environments are volatile and present opportunities that have to be seized and threats that have to be mitigated in order for business to thrive (Kirchmer, 2016). Having high-quality operational processes plays a vital role in the realization of these and other organizational goals (Weske, 2012).

Next to being effective and efficient, processes also need to comply to organizational policies and regulations from external entities. While the abundance of data plays a significant role in the success of many companies, it also brings new laws and regulations to life. This has led to the need for companies to adapt their processes to adhere to these new regulations. Additionally, companies spend considerable amounts of money on internal and external auditing projects to ensure authorities that the necessary controls have been enforced (Ramezani Taghiabadi, 2017). Adhering to all the requirements while remaining effective and efficient can lead to intricate processes. Many organizations use process models to capture the complex behavior for improved understandability. These models can be created using various methods and exist in various forms.

As organizations become more digital, they are starting to record an increasing amount of data regarding their operations (Inside Big Data (2017)., 2017). In contemporary businesses, process executions are frequently recorded to some degree and logged in the information systems of the organization. The availability of this data allows for analysis of their business operations in a bottom-up, data-driven method (van der Aalst et al., 2010). Traditionally, business processes are analyzed by conducting interviews and taking samples of the data. By analyzing empirical process data instead of conducting interviews, factual insights can be generated on the process as a whole. Over the last decade, *process mining* has emerged as a new research field (van der Aalst and Günther, 2007). With process mining, it is possible to use discovery techniques that automatically transform process data into process flows, which can be analyzed based on various objective metrics such as lead time and activity count (van der Aalst, 2011). Process mining also covers another group of techniques called conformance checking. Conformance checking techniques can be used to compare the intended process flow to the generated data of the same process in practice. With this, it is possible to draw conclusions about the conformity of an operational process to its original design, and vice-versa. Methods like these can be of great benefit to process analysts in all fields.

The input required for conformance checking is a process model and an event log (log of all the activities that occurred in the process, retrieved from the information system). In literature, most research on the topic of conformance checking assumes ideal input. However, in practice it is often the case that the event log and process model are not fully reliable (Rogge-Solti et al., 2016). Moreover, the event log and the process model often do not align well (are *incongruous*), due to their different goals. With an event log and a process model of insufficient quality and relative alignment, conformance checking results become meaningless. Little research has been done on the application of conformance checking techniques in practice, where the circumstances are not always ideal. Rogge-Solti et al. (2016) introduce the term generalized conformance checking, which revolves around the application of conformance checking techniques when the input cannot be fully trusted.

This research aims to help close the gap between conformance checking research and practice by applying current conformance checking techniques in a practical setting. Due to lack of reliability in practice, the input needs to be adapted in order to derive meaningful conformance insights. As of now, there is no clear method known to adapt the process model and event log such that they become congruous and therefore suitable for conformance checking. This work attempts to develop a systematic approach to this end.

1.2 Thesis outline

The remainder of this thesis is structured as follows. The rest of this chapter covers the setting in which this research has taken place, as well as the scope of the work. The research problem is elaborated on in Section 1.6, which leads to the research objectives formulated in Section 1.7. After that, the methodology and the running example used in this work are explained. Chapter 2 then elaborates on the underlying concepts of this thesis and aims to provide a common understanding of relevant concepts for all readers. In this chapter, first the current landscape of internal control in modern businesses is discussed. After that, the fundamentals and other relevant aspects of process mining techniques are brought to light. Next, Chapter 3 provides insights into the reasons behind incongruity of a process model and event log, and how this propagates to various problems that need resolving. Chapter 4 then shows how to identify incongruity issues with help of the running example. Resolving the identified incongruities is covered in Chapter 5. This chapter explains the systematic approach developed and the adaptations to be made to the process model and event log. The systematic approach is then applied in the case study in Chapter 6. Finally, the conclusions of this thesis are presented in Chapter 7. A discussion of project limitations and directions for future research is also included in this final chapter.

1.3 Protiviti

This research has been conducted at Protiviti, a global consulting firm based in Amsterdam.

> *"Protiviti is a global consulting firm that delivers deep expertise, objective insights, a tailored approach and unparalleled collaboration to help leaders confidently face the future."* (Protiviti)

Since its inception in 2002, Protiviti has grown to become one of the biggest independent global consulting firms specialized in governance, risk management, and control (GRC), with over 4500 people in more than 70 offices and 20 countries. Protiviti has served over 60% of *FORTUNE 1000* ® companies and has been consistently recognized by *FORTUNE* and *Consulting Magazine* as a best company to work for (Consultancy, 2017). The Dutch office, located in the financial district of Amsterdam, opened in November of 2005. Originally, it was mainly focused on internal audit and risk consulting services. Nowadays, Protiviti Netherlands is an established organization of over 50 people and offers services in all of the global Protiviti solution areas. These solution areas are: *(1)* Internal Audit and Financial Advisory, *(2)* Risk and Compliance, *(3)* Business Process Improvement, *(4)* Transaction Services, *(5)* Technology Consulting, and *(6)* Data Management and Advanced Analytics. If required expertise for a particular project is not readily available in-office, strong collaboration with neighboring offices ensures the project team is always proficient and knowledgeable. With the big data revolution currently happening, Protiviti CEO Joseph Tarantino has made it a worldwide goal to grow in the field of data analytics in the upcoming years.

1.4 Wessanen

The case study of this thesis has been conducted with data of Wessanen. Wessanen is a market-listed multinational food company founded in 1765 and headquartered in the Netherlands. Wessanen has done business in all sorts of foods since then: when it was founded, Wessanen traded in canary seeds. Nowadays, the company has a focus on organic products and is primarily active in Europe (Wessanen, 2018). Half of their revenue stems from their presence in France, where they sell products of major brands such as Bjorg and Bonneterre. In the Netherlands, the company is mostly known for their brands Zonnatura, Clipper, and Whole Earth. In the recent past, Wessanen has launched a company-wide initiative to assess and continuously improve their operational processes.

1.5 Scope

Building upon the introduction of Section 1.1, the aim of this research is to develop a systematic approach for conducting a conformance checking analysis in a practical setting. By using such an approach, business process owners (BPOs) are capable of gaining valuable insights into the conformance of their process executions against the process models describing the desirable process flow. These insights can then be used to further analyze and improve the business process being analyzed, as part of business process management (BPM) initiatives of the organization.

Figure 1.1 shows the BPM Life-Cycle model originally developed by Dumas et al. (2013). The dotted oval in the figure highlights that this research focuses on the sub-process of conformance checking: this is denoted as the connection from the *Process monitoring and controlling* phase to the *Process discovery* phase. This particularly describes *backward conformance checking*, which concerns analyzing past process executions for conformance. Moreover, this work is scoped towards the conformance on a process *flow* level, rather than the underlying event data. This means that the conformance levels are based on how much the activities of a process execution follow the

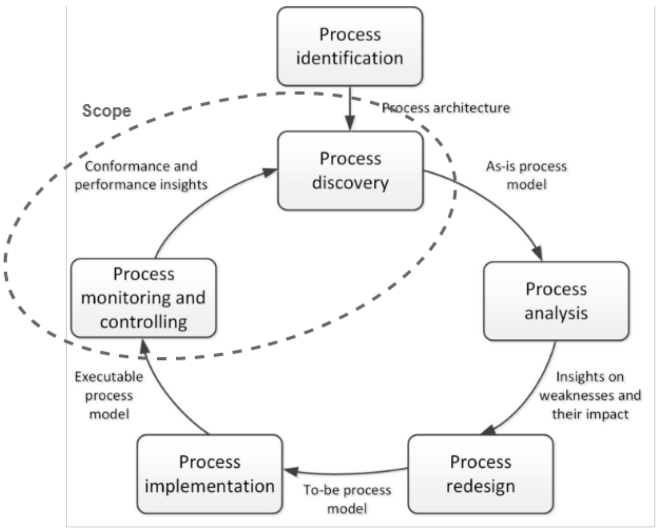


Figure 1.1: Research scope in the context of the BPM life-cycle model (Dumas et al., 2013)

intended path.

1.6 Problem statement

Conformance checking of business processes (referred to as conformance checking in this work) is a complex problem that has been a topic of academic research for several years, as can be seen from van der Aalst (2011), Buijs et al. (2012), and more. While applying conformance checking techniques has proven to be an effective way to compare a process execution to its original design, there exist many caveats to its effectiveness. Most research conducted on the topic assumes that the conditions of a conformance checking analysis are exceptionally good. In this research, all requirements for conformance checking are met and there is little ambiguity regarding the input available. Particularly, the prime inputs needed for conformance checking—an event log and a process model—are typically assumed to be of great individual quality and to match with one another. In other words, the input for conformance checking is assumed to have *congruity.*

> *"Congruity is a quality of agreement and appropriateness. When there's congruity, things fit together in a way that makes sense."* (Vocabulary.com)

Congruity has two key elements in the context of conformance checking:

- The event log and process model *mostly* contain the same activities, on the same level of abstraction (i.e. are *compatible* with each other)
- The event log and process model *mostly* contain those activities that are considered important for conformance checking (i.e. are *relevant*)

Note how these two elements say *mostly.* This is because it is sometimes impossible to have perfect congruity between a process model and an event log. When a process model and an event log are created with different goals in mind, the congruity elements above typically are not present from the start. Although congruity can be improved (as seen in this research), it is sometimes impossible to develop full congruity due to the different natures of these input components. More elaborate reasoning for this is provided in Chapter 3.

As mentioned, research typically assumes that the input for conformance checking is ideal. In practice, the quality of both the event log and process model are not guaranteed to be of high quality. Rather, it is often the case that their quality is not adequate for the purpose of conformance checking. Process models are usually not purposefully created for conformance checking, but serve different purposes (Fahland and van der Aalst, 2015) such as providing operational guidelines to employees or providing a common understanding among stakeholders. Additionally, process models usually are created manually by the organization, which does not always result in a high quality model due to a myriad of reasons, some of which are covered in Claes et al. (2012). Similarly, an extracted event log is not readily useful because it typically lies on a completely different (i.e. lower) abstraction level than the process model (Kalenkova et al., 2016), (Adriansyah et al., 2011). Hence, the process model and the event log are possibly of low individual quality and the combination does not have congruity.

Any data analysis technique requires a suitable input for the results to be reliable and meaningful (often referred to as 'garbage in, garbage out'). The application of conformance checking techniques on unreliable input similarly does not yield significant results. Hence, the problem statement for this research is as follows.

> **Problem statement**: Conformance checking assumes that an event log and the related process model needed as input have congruity. In practice, process models and event logs are incongruous, making them unsuitable for conformance checking.

This work focuses on overcoming this limitation of traditional conformance checking to be able to perform conformance checking techniques on input that is incongruous, as further discussed in the following section.

1.7 Research Objectives

Following from the introduction in Chapter 1 and the problem statement discussed in Section 1.6, research objectives and accompanying research questions are formulated.

> **Research objective**: To develop a systematic approach for performing conformance checking on incongruous input.

The above research objective cannot be reached without answering several smaller questions. It is therefore split into the following research questions:

> **Research question 1**: What are common congruity issues for conformance checking in practice and how can these be identified and explained in the context of the process?
>
> **Research question 2**: Given an event log extracted from a contemporary information system, how can this event log be adapted (improved) such that it has congruity with the corresponding process model and is suitable for conformance checking?
>
> **Research question 3**: Given a process model made available by an organization, how can this process model be adapted (improved) such that it has congruity with the corresponding event log and is suitable for conformance checking?

The first research question revolves around the identification and understanding of congruity issues between the event log and the process model at hand. This is crucial towards solving these issues and creating a congruous pair of an event log and a process model (referred to as a log-model pair in this work). The second and third research questions are concerned with adapting the event log and the process model respectively such that they become congruous with one another such that conformance checking can be performed.

1.7.1 Academic relevance

Answering the research questions above helps research in the field of conformance checking. Most research conducted in this sub-field of process mining assumes ideal conditions (and hence ideal input) to apply particular algorithmic techniques and eventually uncover insights. In order for conformance checking techniques to become more robust and usable in less ideal conditions, there is a need for more research in practice, where the conditions are variable. This work in particular is set out to be a step in the direction of generalized conformance checking (further discussed in Section 2.4), thereby closing the gap between research and practice.

1.7.2 Industrial relevance

For industry, this work makes it possible to use conformance checking techniques on the incongruous input typically available in organizations. This allows process owners, process analysts, process engineers and other stakeholders to analyze how their process performs relative to the process model. Insights gained from these analyses can for instance be used in audit-related activities or to start process improvement initiatives.

1.8 Methodology

To answer the research questions, research has been conducted and experiments have been done. This chapter explains the methodology used to ultimately derive proper conclusions and a systematic approach that solves the aforementioned issues of incongruity.

To develop and apply the systematic approach (also on the final use case), an open standard process called Cross-Industry Standard Process for Data Mining, or CRISP-DM (Shearer, 2000). CRISP-DM is the most widely used and recognized analytics model according to Forbes Forbes (2015).

This iterative process exists of several phases: *Business Understanding*, *Data Understanding*, *Data Preparation*, *Modeling*, *Evaluation*, and *Deployment*.

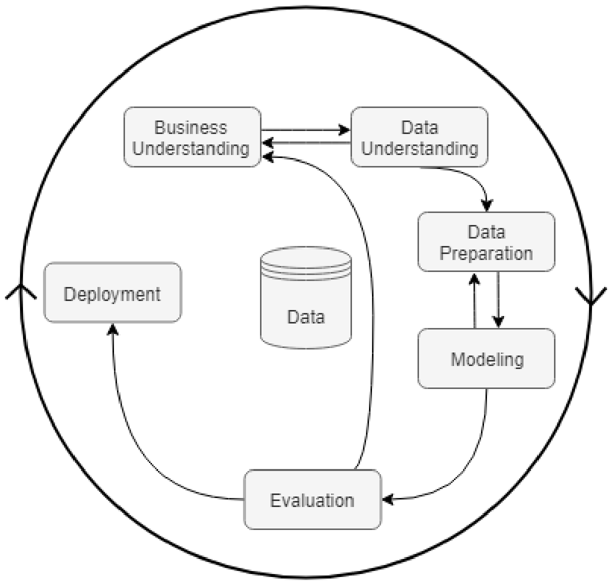


Figure 1.2: Cross-Industry Standard Process for Data Mining (CRISP-DM)

The business understanding phase revolves around grasping the process at hand and the problems that arise within. The goal of the analysis is equally important in this phase. This knowledge is gained by experiencing the process firsthand and speaking to process experts. Data understanding and preparation relates to the inspection and evaluation of the available data, after which it is adapted from the 'raw' data state to the state required for analysis. In the modeling phase, analytical techniques are applied to make models that can be used to generate insights. The evaluation phase naturally aims to see whether the created models provide any meaningful information. The deployment phase is not relevant in this particular work, as this phase is based around implementing the models into an organization. As this work is mostly research, deployment is not needed. The iterative approach of CRISP-DM allows for relatively fast feedback loops, which are needed to come up with quality results. More elaborate details on how each phase is handled can be seen in Chapters 3 through 7.

The CRISP-DM process was used to derive the systematic approach and is used to showcase how it applies on the running example discussed in Section 1.9. The CRISP-DM approach is also applicable for the systematic approach, which was applied on the use case. For creating and applying the systematic approach, a combination of tools was used. Replay conformance checking techniques are performed in open-source process mining tool ProM, whereas the whitelisting approach (mentioned in Chapter 5) is used in Celonis, a market-leading commercial process mining tool. Chapter 6 will explain these techniques in more detail.

1.9 Running example

Throughout this work a running example is used to explain certain relevant concepts related to the incongruity of a log-model pair. The running example also acts as a guidance throughout the introduction of the systematic approach, providing a more tangible understanding of what steps are performed in the approach. The running example (also referred to as example hereafter) consists of the two prime input elements for conformance checking: an event log and a process model. They describe the same imaginary insurance claim process, which is an example that is often used in process mining literature. In short, the process goes as follows. First, the ticket is received by the insurance company, after which the request is registered in the system. Then, the ticket has to be reviewed to see whether it adheres to all the requirements for a legitimate ticket. If the ticket is considered fine, the ticket is investigated and a decision is made on the potential compensation. If not, a new ticket is requested. After rejection or a compensation paid, the request is closed.

In the case of this work, the example used does not have congruity. The event log in the example was created for the sake of process discovery: creating a process model from operationally generated event data (explained in Chapter 2). The process model on the other hand resides on a higher level of abstraction and provides an understanding of the process to upper management of the business. This results in two components (event log and process model) with process steps that do not necessarily match. Hence even though both components describe the same process, the log-model pair does not form a congruous match. This makes the pair unsuitable for conformance checking as-is. Reasoning for discrepancies between event logs and accompanying process models is further explained in Chapter 3.

The following shows a small excerpt of the event log used as the running example. The excerpt is a single trace that is considered to have no real nonconformities (i.e. it fully conforms to the intended process). In this particular case, the request is registered, after which the ticket is scanned and goes through Optical Character Recognition (OCR) to automatically generate information. Then, the ticket is reviewed for completeness and posted to the system for complete tickets. Pete is the person that investigates the request and approves it. The compensation is then paid and the ticket is closed by Bob. The full event log can be found in Appendix A.

Table 1.1: Excerpt of event log used as running example

Case ID	**Activity**	**Timestamp**	**Resource**
17	Register request	20-09-2018 10:12	Bob
17	Scan ticket	20-09-2018 10:14	Bob
17	OCR on ticket	20-09-2018 17:12	System
17	OCR complete	20-09-2018 17:12	System
17	Sent for validation	20-09-2018 15:45	System
17	Validation complete	23-09-2018 09:12	Claire
17	Post ticket	23-09-2018 09:15	System
17	Approve request	26-09-2018 12:02	Pete
17	Pay compensation	31-09-2018 00:00	System
17	Close ticket	01-10-2018 09:15	Bob

The process model that is used in the running example describes the same process, but the activities in the model do not contain many commonalities with those in the event log. Figure 1.3 shows an excerpt of the model. The full process model can be seen in Appendix A.

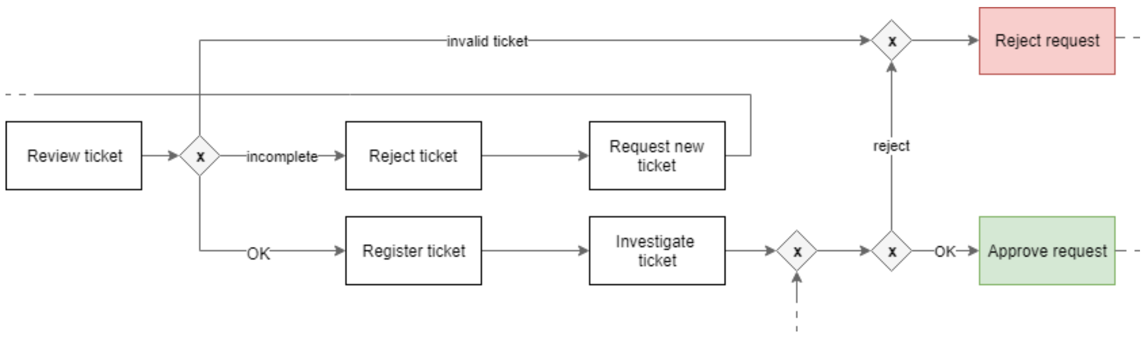


Figure 1.3: Excerpt of process model used as running example

The excerpt of the process model shows the part of the process in which a ticket is reviewed for completeness and investigated for rejection or approval. This part of the process starts with the 'Review ticket' activity, which is the completeness check. If the ticket is incomplete, a new ticket is requested with the 'Request new ticket' activity. The ticket can also be rejected altogether if it is considered invalid, through the 'Reject request' activity. If the ticket is complete, the next activity is 'Register ticket'. After the ticket has been registered, the ticket is investigated ('Investigate ticket') to determine whether compensation should be paid ('Approve ticket') or not ('Reject ticket').

The incongruity between the two components can be identified by looking at the activities in the process model and comparing them to the events from the event log. Although describing the same process, the two components have no common names for the process steps. This is because the process model describes the activities that occur in the process on a higher level of abstraction for process understanding. On the other hand, the event log contains recorded events that have occurred in the information system. These are on a lower level of abstraction. Because the log-model pair do not have much in common (are incongruous), performing conformance checking would result in useless results.

This example (i.e. the combination of event log and process model) is used as the basis for the systematic approach to solve incongruity between event log and process model. Throughout this work, the running example will be referenced and adapted to show that it is possible to perform meaningful conformance checking on incongruous input.

Chapter 2

Theoretical Background

This chapter provides a common base of understanding by describing the relevant concepts for this thesis. Throughout this chapter related work is discussed that is used as an academic basis for this research. First, Section 2.1.4 explains the concepts of conformance, compliance and control. Then, Section 2.2 covers the fundamentals of process mining itself. Conformance checking is then handled in Section 2.3. Finally, this chapter includes an introduction to the concept of Generalized Conformance Checking, which is included in Section 2.4.

2.1 Conformance, compliance, and control

In order to fully comprehend the need for conformance checking in practice it is important to realize why it may be critical to follow a particular business process flow. To this end, the concepts of conformance, compliance, and control are elaborated upon in this section.

2.1.1 Definitions

Conformance, compliance, and control are terms that are often wrongly interchanged in practice Flinders and Denton (2008). Nevertheless, the distinction between these terms is important to understand the need for (measuring) effective process conformance. For each of these terms, the definitions that are used in this work are given below.

- *Conformance*: In accordance with a standard or guideline. An operation *should* follow a particular standard (e.g. adhering to an ISO-9001 standard). In the context of this work, a process execution *should* conform to the process model that describes the desirable behavior of that process.

- *Compliance*: Meeting the requirements of a specific regulation or law. An operation *must* follow rules set by external entities (e.g. adhering to the General Data Protection Regulation). In the context of this work, a process execution *must* follow a particular flow because of regulations (this must be visible for external auditors to check).

- *Control*: A control is set in place by an organization to provide assurance regarding the objective of some goal (e.g. separation of duties on tasks that should be performed by two different individuals in order to prevent fraud).

These concepts work together in many organizations. Business processes are designed such that they are compliant to legislation. At the same time, processes are designed and implemented to be effective and efficient, according to some predefined metrics. The next section will elaborate on the need for processes that closely follow the designed processes.

2.1.2 Conformance and Compliance

Conformance and compliance are words that are often used interchangeably. Although their meanings can be similar in context, the definitions above have clarified that they have different meanings.

In terms of of the design of business processes, however, the terms are quite similar. Both compliance to legislation and conformance to particular standards result in the design of a business process that adheres to said legislation or standard. When a business process has been designed and systems have been set in place for operations to follow this process, the two terms become somewhat ambiguous, as the process is supposed to adhere to the design. When it comes to implementing systems and internal control measures, it is still pertinent to distinguish the two terms, as it determines the amount of effort that should be put into *ensuring* that the process is conforming to the designed process (and hence, compliant to legislation). In this work, *conformance* to a designed process is used as a leading term. Moreover, the term *nonconformity* is used to describe a part in a process execution that deviates from the intended process.

The designed process may or may not include elements that stem from legislative *compliance*, but these will be discussed as regular process design elements that an operational process should conform to. Nonetheless, references are made to regulatory compliance to address the importance of process conformance and give context to the issue at hand.

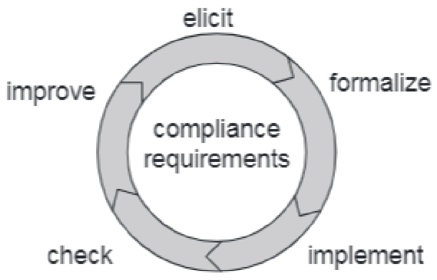


Figure 2.1: Compliance Management life cycle ((Ramezani Taghiabadi, 2017))

To ensure that processes adhere to legislation, there is a five-phase Compliance Management Life Cycle developed by Ramezani Taghiabadi (2017), seen in Figure 2.1. This life cycle is equally applicable to general conformance checking, and is set up as follows. The (1) *compliance elicitation* phase is used to determine the important constraints that a process needs to adhere to. Then, the (2) *compliance formalization* phase is to derive precisely formulated constraints from the results of the compliance elicitation phase. After that, the (3) *compliance implementation* phase revolves around implementing and configuring the systems such that the process adheres to the formalized constraints. Next, the (4) *compliance checking* phase aims to investigate both whether the formalized constraints will be met in the future, and whether the constraints have been met in the past. This relates to the two types of conformance checking discussed hereafter. Lastly, (5) *compliance improvement* is the final phase, in which the process design and implementation is improved based on diagnostic information retrieved in the compliance checking phase. In the context of this life cycle, the focus of this work lies in the fourth phase (i.e. compliance checking). The following will elaborate on this area of focus.

Broadly, there are two classes of conformance checking: forward conformance checking and backward conformance checking. Simply put, this classification determines whether the approach is preventive (x should or should not happen) versus detective (this is what happened). This classification is originally meant (by ...) for compliance checking, but again, is just as applicable for conformance checking in general. The two classes are described as follows.

- *Forward conformance checking* aims to design, implement, and execute processes that conform to the desired behavior. This includes techniques to identify desired behavior, to design a process such that it follows that behavior, and to ensure that a process in execution conforms to the design.
- *Backward conformance checking* aims to detect behavior that is not conforming in hindsight. Backward conformance checking techniques make use of execution data to find nonconformities. By evaluating process executions, it is possible to see whether rules were adhered to, and where and when violations occur. These techniques are unable to prevent nonconformities from occurring during a process execution.

Forward conformance checking techniques can be performed without any previously generated data, as these techniques are to be implemented for live process execution. Conversely, backward conformance checking techniques need the process to have generated some event data that can be analyzed. To further narrow the scope of this work, the focus in this research is on *backward* conformance checking (hereafter simply conformance checking).

2.1.3 Constraint dimensions

There exist many different constraints that can be placed on a business process (Ramezani Taghiabadi, 2017). Section 3.1 defines the dimensions of conformance constraints as being the following:

- *Control flow* constraints revolve around the standard sequence of activities in a process, for any case instance. For example: "After a claim is filed, validity must be checked."
- *Data flow* constraints are related to flow of activities as well, but depend on the event data associated with the particular case instance. For example: "A claim of over 3000 euros requires two validity checks."
- *Organizational* constraints impact the method in which a process is executed. For example: "Multiple validity checks are carried out by different employees."
- *Temporal* constraints relate to the time it takes to perform certain activities. For example: "The claim must be processed within 6 months."
- *Model-based* constraints say that a process must include certain elements for control. For example: "The claim process must have a time-out handler."
- Next to constraints regarding a single process execution there can also exist constraints on a group of process executions. An example of such a *multiple case* constraint: "20% of claims require a detailed check."

A process constraint may be limited to an individual constraint dimension or may be a combination of multiple. As an example: "After a claim of more than 3000 euros has been filed, two different employees need to check the validity of the claim independently. Each claim must be handled at most 6 months after the placement." As can be seen from the above, there exist several different constraint dimensions and the possibility to combine these, makes it difficult to develop an approach that covers all dimensions. The approach introduced in this work will limit its focus to the control flow constraint dimension. Having a process with good control flow standard is considered the basis for an effective, efficient, and controlled process. Constraints such as the ones above are used to implement such a process.

2.1.4 Internal Control

Internal control relates to the effectiveness of the business process at hand, including the constraints implemented. Internal control is defined by the COSO framework (COSO, 2013) as:

> "Internal control is a process, effected by an entity's board of directors, management and other personnel, designed to provide reasonable assurance regarding the achievement of objectives in the following categories:
>
> - Effectiveness and efficiency of operations
> - Reliability of financial reporting
> - Compliance with applicable laws and regulations"

Although the above categories are distinct, they do not necessarily divide the controls into distinct categories; overlap may exist. Each category addresses a different goal for the organization at hand. Naturally, the goal for the first category is a basic business objective: to have effective and efficient operational processes. The second category of control aims to create high-quality financial reports that present reliable numbers. This can be very beneficial for the organization itself for various reasons (effective leadership, identifying opportunities), but may also be mandatory for external organizations in various circumstances (market, legislation). The third category deals with organizations having to comply to specific laws and regulations.

Note that the definition given by (COSO, 2013) says "*reasonable* assurance", rather than an implied absolute assurance. This implies that it is sometimes impossible to create complete assurance, but that a degree of assurance should be provided. This degree of assurance depends heavily on the situation at hand and can be constrained by costs and the potential benefit that additional control policies and procedures may generate.

The reasonable assurance to be provided comes from five interrelated components of internal control, mentioned by Ratcliffe and Landes (2009). The first is the *control environment*, which is there to serve as the basis for all other components. It relates to the environment that is created by management, including practices and actions that support the organization's values. Then, there is *risk assessment*, relating to the identification and analysis of risks, both in reporting and in operations. *Control activities* exist to make sure that the directives set by management are carried out. Activities exist of policies and procedures that decrease the risks towards achieving the goals of the organization. Typical examples within this component include authorization, approvals, reviews and segregation of duties. Next, *information and communication* plays a role in how information travels through an organization. Many processes are supported by systems that record information. Additionally, internal communication through email and meetings play an integral part. Lastly, *monitoring* of internal control systems is pertinent to assess the quality of the control activities over time. This monitoring should be done on an ongoing basis, possibly combined with separate evaluations. Of these five interrelated components, the monitoring component is where automatic detection of process nonconformities should take place, as this is where the quality of the process design is assessed.

2.1.5 Questions to answer

Once the controls have been designed by the organization, enforced by the supporting systems, and embedded into the culture, process executions should follow the modeled process flow. Unfortunately, exceptions occur and processes do not always follow the desired flow. This could be due to human error, poor implementation, exceptional circumstances, or other reasons. [source] In order to reduce the number of deviating process executions, it is important to know what exactly is happening that is causing the nonconformity. To this end, several questions should be answered. To start with, some objective facts that can be retrieved directly from associated data should be gathered by answering the following questions (and possibly more):

"*What are the violated constraints? How many times was the constraint violated? Which activities play a role?*"

After that, analysis can be done on the more in-depth reasoning behind the nonconformity, by asking questions such as:

"*What went wrong exactly? What should have happened instead? Which violations are most important and require a closer look first? What are the causes of a specific violation? Who are the actors involved? Is there an indicator in the process that allows us to predict future violations? Can we find a pattern?*"

Some of these questions could be relatively simple to answer by interviewing a process owner, while others may require some extensive research in collaboration with the organization. This work aims to create an approach that allows an organization to gather the answers to the first type of questions: more objective, data-driven questions. The answers to these questions are ingrained in the process data, but are not always simple to retrieve, due to a myriad of reasons discussed in Chapter 3.

2.2 Process Mining

This section provides a basic understanding of what Process Mining is in Section 2.2.1. This section also explains its role in the context of organizations. Special attention will be given to the event log, which is of additional importance in later chapters. This is done in Section 2.2.2

2.2.1 Overview

Process Mining is a family of data analytics techniques that can be used to provide insights into business processes. Many business processes are supported by contemporary information systems (workflow management systems, enterprise resource planning systems, customer relationship management systems, traditional database systems) that record and/or store particular process-related events. Selected data can be extracted from these information systems to create a so-called *event log*, which is the prime input for all process mining techniques. Using said techniques then provides the user with the ability to *"discover, monitor, and improve real processes"* (van der Aalst, 2011).

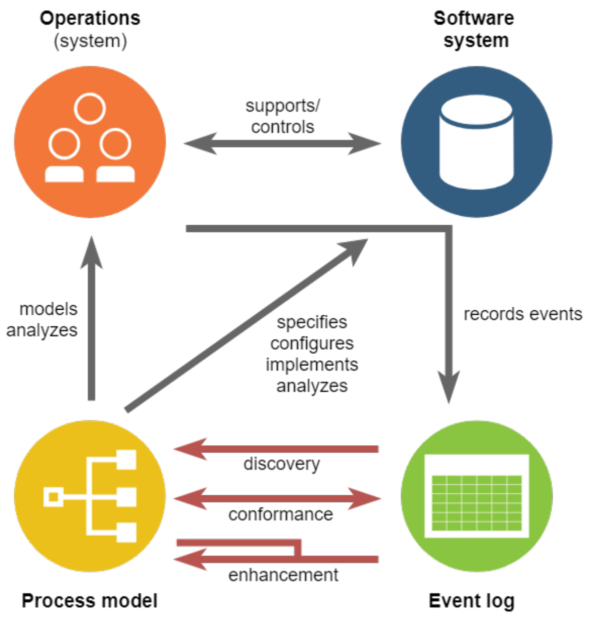


Figure 2.2: Process mining overview (visual adaptation from the Process Mining Manifesto)

Figure 2.2 provides an overview of the positioning of process mining in the context of the organisation (IEEE Task Force on Process Mining, 2011). The three red arrows in this figure denote the three main types of process mining techniques. *Process discovery* can be used to automatically develop a process model based on the event data, without requiring any prior information. Such a model can then be used to explore (discover) the functionality of the operational process. The second type, *conformance checking*, can be used to validate how well a process model describes the execution of the actual process, and vice-versa how well the process execution follows the model. Third, *process enhancement* aims to improve an existing (human-made) process model by using information about the operational process. As aforementioned, all three main types of process mining require an *event log* as an input. Because the focus of this work is on conformance checking, there is also need for a secondary input: a process model. The next section describes the structure of such a log, to provide an understanding of the foundational concepts of process mining.

2.2.2 Event log

An event log is a collection of all the activities that occur in a particular business process. Information gained from (process-aware) information systems can be extracted to create a log that describes particular process instances. The log consists of different cases that each have their own sequence of events, which is typically called a *trace*. An example trace for an arbitrary case X could be $\langle a, b, c \rangle$, where a, b, and c are events in the set of all possible events. An event log holds several (possibly many) of these traces, so an event log can be denoted as a set of traces, such as: $L = (\langle a, b, c, d \rangle, \langle a, c, b, d \rangle, \langle a, d \rangle)$. Each event is annotated with a timestamp, which makes it possible to sort individual events on their moment of occurrence. Furthermore, events can be enriched with additional information—such as the resource used or an event cost—for deeper analysis. Because event logs can become quite packed with information, they are often represented as a table, as seen in Table 2.1.

Table 2.1: Example event log of a simple imaginary compensation request process

Case ID	**Activity**	**Timestamp**	**Resource**	**Cost**
1	Register request	20-09-2018 10:12	System	0
1	Check ticket	20-09-2018 10:14	Pete	100
1	Decide	20-09-2018 17:12	Sarah	150
1	Reject request	21-09-2018 09:18	Mike	50
2	Register request	20-09-2018 15:45	System	0
2	Check ticket	23-09-2018 09:12	Ellen	100
2	Decide	23-09-2018 09:15	Sarah	150
2	Pay compensation	26-09-2018 12:02	Mike	200
3	Register request	24-09-2018 11:12	System	0
3	Check ticket	25-09-2018 13:15	Pete	100
3	Cancel ticket	25-09-2018 13:20	Pete	50

The event log above shows three different traces for an imaginary claim handling process, with three separate case identifiers (1, 2, and 3). Every row in the table represents one event, described by an activity, a timestamp indicating when the event occurred, the resource that performed the activity, and an associated cost. Only the first three columns—the case identifier, activity name, and a timestamp— are necessary for process discovery: the automatic generation of a process model that allows for the behavior seen in the event log. Any additionally annotated information, such as the resource and cost in Table 2.1, is not necessary for process discovery but adds information that can be relevant for analysis. For example, analyzing certain internal control policies such as segregation of duties is only possible if the user accounts associated with the activities is included in the data.

Generating an event log may require very little effort depending on the nature of the supporting information system and the process-awareness maturity of the organization. If the information system is a workflow management system in which all process steps are already logged and structured in a way similar to what is needed, generating an event log is very straightforward. Things become more complicated when the process is supported by a scattered system landscape where (some) single activities are recorded, but not in the context of a process. Combining the right data sources and incorporated activities may require some more technical affinity and knowledge of the different components within the landscape.

2.3 Conformance checking

This section discusses the topic of conformance checking, first giving a general overview in Section 2.3.1. Then, metrics used are covered in Section 2.3.2 and the various conformance checking techniques available are discussed in Section 2.3.3.

2.3.1 Overview

Conformance checking is a type of process mining techniques that can be used to validate how well a given process model describes the execution of the actual process in practice (van der Aalst, 2011). Conversely, it also gives insights into the reverse: how well does the execution follow the—likely desired—flow of the process model? For conformance checking to be performed, there is need for two inputs: an *event log* and a *process model*, as shown in Figure 2.3. The event log has already been discussed extensively in Section 2.2.2. The process model used as input can either be human-made (conceptual) or derived from process discovery (data-driven).

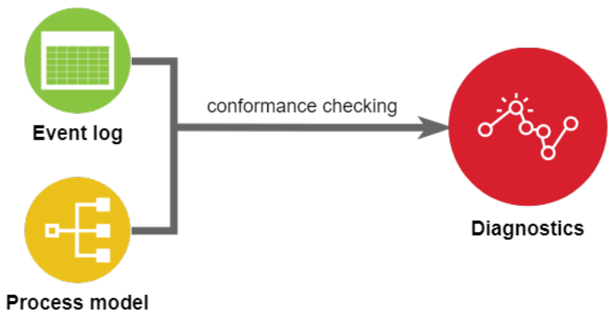


Figure 2.3: Conformance checking input and output

When conformance checking is performed, the observed behavior is compared to the modeled behavior to find commonalities and discrepancies as an output. For example, conformance checking can reveal that the operational process deviates from the model where two activities should be performed in sequence (i.e. the event log includes only traces with $\langle .., a, b, .. \rangle$) according to the model, but the activities are also performed the other way around (event log includes traces with $\langle .., b, a, .. \rangle$). From another perspective, it could be that the model allows for more behavior than what actually occurs in practice (i.e. the model allows for traces with $\langle .., a, b, .. \rangle$ and $\langle .., b, a.. \rangle$, but the latter never occurs in the event log. The nonconformities above could respectively indicate that the process executions include traces that do not fit the model at hand, and that the model is incapable of correctly describing the recorded behavior. The interpretation of these nonconfirmities largely depends on the model used.

Nonconformities can be interpreted in different ways depending on the purpose of the model and the type of discrepancies between the model and the log. If the process model used describes the intended process flow (i.e. is *normative*), discrepancies between the two inputs indicate that the process does not follow the flow that it should. This in turn means that the operational process needs to be adapted to the wishes of the organization. If the process model describes the process as it is executed in practice (i.e. is *descriptive*), discrepancies show that the model does not fully capture all the possible sequences of steps in the process. Another possibility is that the model is descriptive, but only of the so-called *happy path* or *happy flow*. This means that it provides a description of the process as it occurs when there are no exceptional conditions. In this case, nonconformities indicate that there are indeed exceptional (possibly undesired) conditions.

The diagnostics found as an output may not only have to do with effectiveness and efficiency of the process flow itself but also with internal control measures related to risk, fraud, compliance, and possibly more. Insights can be given into the effectiveness of control measures, such as a segregation of duty control on two subsequent activities. This can be done using an event log that has been augmented with additional event data.

2.3.2 Conformance metrics

The commonalities and discrepancies are not just observable in a qualitative way, but can also be expressed by means of several metrics (van Dongen et al., 2016). These metrics are a way to express the overall conformance of the process model and the event log. The most common ones are discussed below.

- *Fitness*: Used to determine the amount of event log behavior that is accounted for in the process model. If a model has a fitness of 100% (perfect fitness), it means that every single trace in the event log could be replayed correctly on the process model. In short, the fitness indicates the ability of the process model to explain the observed behavior in the event log.
- *Precision*: Evaluates how much behavior that is not in the event log is producible by the process model. Precision is an important metric, because a model that is too general will allow for behavior that is not actually permitted. This in turn means that the model becomes less informative, as it describes more than the actual process. An extreme example often used in research is that of the flower model: a model that allows every activity to be followed by any other activity. Although this model allows for every trace (giving it a fitness of 100%), it has a very low precision because it allows for a lot of behavior that is not found in the event log. This dimension is concerned with the avoidance of underfitting (i.e. allowing for too much unintended behavior).
- *Generalization*: Because a model must be able to reproduce possible future behavior, it should not be overfitting on the traces found in the event log. This metric is used to determine how much the model is overfitting (i.e. allowing for strictly the behavior seen, but possibly not other intended behavior).
- *Simplicity*: The simplicity metric determines how well structured a process model is. If the model contain any unnecessarily complex structures such that it hinders proper understanding of the model, this decreases the simplicity metric. The simplicity dimension is often explained with *Occam's Razor* (i.e. the simplest explanation is often the best).

While fitness and precision are mathematically calculated and are hence strict, the generalization and simplicity dimensions are quite subjective. Depending on knowledge of the process at hand, two individuals may have very different assessments of generalization and simplicity on a single process model. Although good indicators of conformance, this work does not necessarily strive for any of these dimensions to be as good as possible. This is because even though a high level of fitness can be achieved on a technical level, the aim of this work is to provide meaningful insights regarding the process and not necessarily have a high fitness level based on technicalities. Nevertheless, the fitness and precision dimensions will be taken into account in the following chapters.

2.3.3 Techniques

There exist several main types of conformance checking techniques that each have their own strengths and weaknesses (Buijs et al., 2012): log replay, trace alignment, and behavioral alignment. It is important to note that all the techniques mentioned focus on control-flow conformance, which is what this work focuses on. Data-flow conformance and additional elements such as distribution of work are out of scope in this work.

Log replay techniques work by trying to execute each event in a trace on a process model, to see whether the trace *conforms* to the model. Whenever an error is found, the techniques make a local correction by making a imaginary step in the event log or the process model in order to come to a proper completion of the trace. The result of these techniques is typically an overview of log-model fitness, as well as a per-trace view of how the replay was done, including the steps taken to overcome the errors.

Trace alignment techniques attempt to find the producible trace in the process model that is closest to a particular trace in the event log (van der Aalst et al., 2012). The output for such techniques is a set of pairs of aligned traces in which every pair includes one trace in the event log that does not fit on the process model, and the closest trace that *is* producible on the model. This type of technique is not very suitable for conformance insights into concurrent behavior, due to the fact that traces are individually handled. For example, suppose there is a trace $\langle .., a, b, .. \rangle$, and another trace $\langle .., b, a, .. \rangle$. Now assuming that the process model does not describe concurrency between these two activities, the trace alignment technique does not provide an overall insight about the lack of concurrency, because the two event log traces are handled individually. Therefore, there is a limitation to these techniques: the insights they provide are on a local trace level, rather than a global level.

Behavioral alignment techniques also generate an alignment between the event log and the process model, but on an overall level. The complete state space of the event log (i.e. all possible traces) is aligned against the state space (i.e. all possible process executions according to the model), and states are detected in which there are discrepancies.

More techniques exist in which traces in the event log are supplemented with *negative events* that do not actually belong in the log. For example, if the event log shows that no traces $\langle a, b, .. \rangle$ are followed by an event c, we can insert a fake event such that we get a log with $\langle a, b, c, .. \rangle$. When replaying the event log on the process model, it is possible to see whether the model accepts this additional negative behavior, in which case the model does not correctly capture the operational process flow.

2.4 Generalized conformance checking

Conformance checking can be greatly beneficial to many organizations to find nonconformities in their process executions or to find that their models do not properly resemble the real world. With modern organizations making an effort to digitize their operations, process models and event logs are becoming increasingly available. However, it is often the case that the event logs and process models retrieved do not have enough congruity to directly generate meaningful insights by means of (traditional) conformance checking. As explained in chapter 2.2, the event log could be used to perform *process discovery* and create a process model that has congruity with the event log. Doing this, however, discards any value that the original model—created manually by the organization—has.

To solve the reliability issues on the side of process models, Fahland and van der Aalst (2012) introduces *model repair.* In this work, the authors explain how to repair parts of a process model that do not align with the matching event log, such that the model-log pair have congruity and the value of the original model is preserved. With all techniques proposed, it is important that the repaired model stays as close to the reference model as possible. In this work, the event log is understandably used as an absolutely truthful source, because it is based on empirical evidence. Although this makes sense from a purely technical conformance checking perspective, the event log cannot always be fully trusted, due to a myriad of reasons further discussed in Chapter 3.

In the context of this work, both the process model and the event log cannot be fully trusted. The authors of Rogge-Solti et al. (2016) introduce the term *Generalized conformance checking* to to overcome the issue of having input that is not fully reliable. Generalized conformance checking refers to more than just identifying discrepancies (and commonalities) between a process model and an event log, as it also categorizes the nonconformities to give an additional understanding of the issue. The authors differentiate between three distinct categories:

1. Anomalies in the event log
2. Modeling errors
3. Unresolvable inconsistencies

The first is *anomalies in the event log.* These are issues in an event log that make the event log less trustworthy. The second category is *modeling errors.* Similar to event log anomalies, these are problems in the process model that cause it to be less trustworthy. These errors can be repaired with model repair techniques discussed prior. Third and last, there is a category for *unresolvable inconsistencies.* Examples for all tree categories are given in Chapter 3.2.

> “Generalized conformance checking unites the three tasks of model repair, log repair, and conformance checking under a common roof.” - Rogge-Solti

This work aims to provide a step towards generalized conformance checking, by introducing a systematic approach that allows the user to create a congruous log-model pair from input that is not fully reliable. This systematic approach contains both log repair and model repair techniques and requires a third input that is formally introduced in the following chapter.

Chapter 3

Incongruity: Causes and Context

Knowing that an event log and the accompanying process model do not have congruity, it is necessary to identify the issues at hand to be able to overcome these and perform meaningful conformance checking. This chapter first describes the causes for log-model incongruity in Section 3.1, which helps understand the actual congruity issues. The subsections 3.1.1 and 3.1.2 discuss the individual quality of the event log and the process model. After that, Section 3.2 provides an overview of the possible congruity issues that can be found in practice, explained within their context for proper understanding. An explanation is given on how to identify these with the help of the running example presented in Chapter 1.

3.1 Cause of Incongruity

To fully understand the congruity issues at hand, it is important to understand the underlying relationship between the process model and the event log that causes the incongruity in the first place. Figure 3.1 shows an overview of the four different levels relevant in the creation of a process model and an event log in the form of a class diagram. This figure is an adaptation of a diagram used in the work of van der Aalst (2011). The figure describes four different levels (dimensions) in the full relationships between a particular business process, a process model, and an event log. The edges between the individual components describe the relationship between components. These relationships are elaborated upon below.

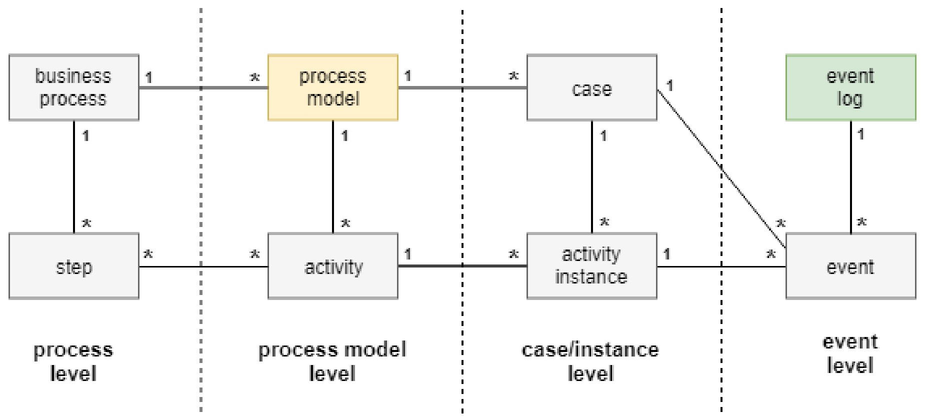


Figure 3.1: Business process to event log class diagram (adapted version from Process Mining book)

The eight components found in the figure are introduced in *italic*. The relationships between the different components are discussed, as well as the degrees of relationship (cardinality). From left to right, the different levels can be explained as follows.

- The **process level** is where the actual operational business process takes place in practice. A *business process* is executed by means of different *steps* in the process. The $1 : *$ relationship between these two components is explained as: one business process consists of a multitude of different steps and each step belongs to a single business process.

- The **process model level** is the modeled dimension of a business process. The *process model* component is a modeled version of the real business process and exists of a number of different *activities*, hence the $1 : *$ similar to the process level. Between the *business process* component and the *process model*, there is another $1 : *$ relationship due to the fact that a single business process may be modeled in different ways for different purposes, as discussed in Chapter 2. The relationship between a process *step* and an *activity* is of degree $* : *$, because a step in the process may be modeled as multiple activities in a model. Conversely, multiple process steps may be modeled as a single activity (e.g. small steps with little individual significance modeled as one larger activity).

- The **case/instance level** covers the execution of a particular process instance in the process model. The leading component in this level is the *case*, that exists of several *activity instances*, explaining the $1 : *$ relationship. One *process model* can have multiple different *cases*, which in turn explains the $1 : *$ relationship there. The same holds for *activities* and *activity instances*.

- The **event level** relates to the information system and its traces representing the business process. By using information from the *case* and the *activity instance*, it is possible to create the *event* component. The event holds information regarding a particular activity instance in a case. One case holds multiple events, hence the $1 : *$ relationship. At the same time, one activity instance can translate to multiple events in the system, resulting in another $1 : *$ relationship between those components. All of the events related to a single process are grouped in an *event log*, explaining the $1 : *$ relationship there.

In figure 3.1, the process model and event log components are highlighted in yellow and green respectively. The diagram shows that the process model level and the event level are not directly connected, but have the case/instance level as a connecting level. The process model describes the flow of a particular process instance, and similarly, the flow of the process instance generates the data in the supporting information system that is eventually turned into an event log. The different levels they are in, the levels in between them, and the relationships between the components are multiple reasons to conclude that there *can* exist significant discrepancies between the two key inputs.

Additionally, there is another reason why the process model and the event log cannot always have congruity. Although the process model level and the event level describe the same process, the process model and the event log often have very different functions. The event log is created for the sake of process mining (process *discovery* in particular), and is formed by extracting event data from the information system that supports the business process. On the contrary, the process model is typically not created with the same goal. Rather, the model is often manually created for (and possibly by) the business process owner (BPO), who is responsible for managing and monitoring the process. A process model can help form a better understanding of the process at hand, so that the BPO can achieve their goals more effectively. This difference in function is also reason for the event log and process model to not always form a congruous match, because the different goals make for different modeling elements (other activities, structures, naming conventions). The next subsections discuss the individual quality of the event log and the process model in the context of process mining.

3.1.1 Event log quality

The event log is the prime input for any process mining technique (not just conformance checking). As previously explained, the event log is comprised of many recorded events retrieved from information systems supporting the business process. Luckily, the events recorded by the system do not have to reside in dedicated log files per se. Events can be stored in any way: be it in structured database tables, transaction logs, loose messages, or any other data structure. The data structure in which the events are recorded does not necessarily impact the quality of said event data, but rather the effort needed to create a proper event log of the recorded events.

As with any data analysis, the quality of conformance checking results is heavily dependent on the quality of the input. Unfortunately, not all information systems are built to store event data during all steps of a process. In the best case, workflow management systems securely store every single step along the workflow, with detailed information regarding every step. On the other hand, for some organizations the only 'system' that may be in place is a paper trail of sticky notes, describing some but not all of the events in the process with very limited (sometimes missing) information. Naturally, the way in which the process steps are recorded heavily impact the quality of the event log that can be created.

The Process Mining Manifesto IEEE Task Force on Process Mining (2011) mentions several criteria to judge the quality of event data. Arguably the most important criterion is that the event data is reliable, meaning that the events recorded have actually happened and that the associated information is correct. Second, event data should be complete. No events that occurred should be missing from the record, given a particular process. Then, event data must have defined semantics: all event records should have a similar structure. Lastly, security and privacy concerns regarding the data should be addressed and considered safe. Based on these four criteria, the Manifesto introduces a maturity model consisting of five different star levels, which can be found in Appendix B. The Manifesto suggests that the event log must be at least at the third maturity level ($\star\star\star$) for process mining to be effective. Although possible, performing process mining on event logs with level $\star$ and $\star\star$ is considered to not provide trustworthy results.

The event log used as a running example throughout this work is considered to be of level $\star\star\star\star$, which is the second-highest maturity level. The reason for this classification is that the event log was retrieved directly from the information system, including a clear process instance (case) indicator. Moreover, the recorded events are trustworthy (reliable) and complete (there are no missing events). *Note that these claims normally have to be checked within the information system and with the process owners.* However, nothing can be said about the security and privacy concerns, and there are not many additional attributes that provide ontologies (i.e. descriptions of relationships between process elements). These elements are needed for a log to be of level $\star\star\star\star\star$. Because the log is of this quality level, it is safe to say that it can be used for process mining. Nevertheless, this does not mean that the event log is issue-free. The event log is of reasonably high individual quality but that does not mean that the event log and the process model have congruity, needed for conformance checking. The event log may still require adaptation towards this end, which is discussed in Chapter 5.

3.1.2 Process model quality

For conformance checking, not just the quality of the event log is important but also that of the process model. Even if the event log is of perfect quality, conformance checking will not give any good results if the process model is of terrible quality. For this reason, it may be valuable to assess the quality of the process model itself before attempting to perform conformance checking. Unfortunately, a similar maturity model to that of event logs does not exist. Because process models can vary so much due to different functions , abstraction levels, and semantics (Claes et al., 2012), it would be very difficult to create a generalized maturity model for process models. However, it is possible to classify process models based on their semantics. This gives an insight into the level of formality a model has. Although this is not directly relatable to process model quality, the formalization of models can be important as some conformance checking techniques (such as replay techniques used in this work) require a model that can replay traces from an event log. Models with a lower level of formalization do not allow this. Examples of the different formality levels can be seen in Figure 3.2. The figure makes use of excerpts from the example model used throughout this work. The different formalization levels are explained below.

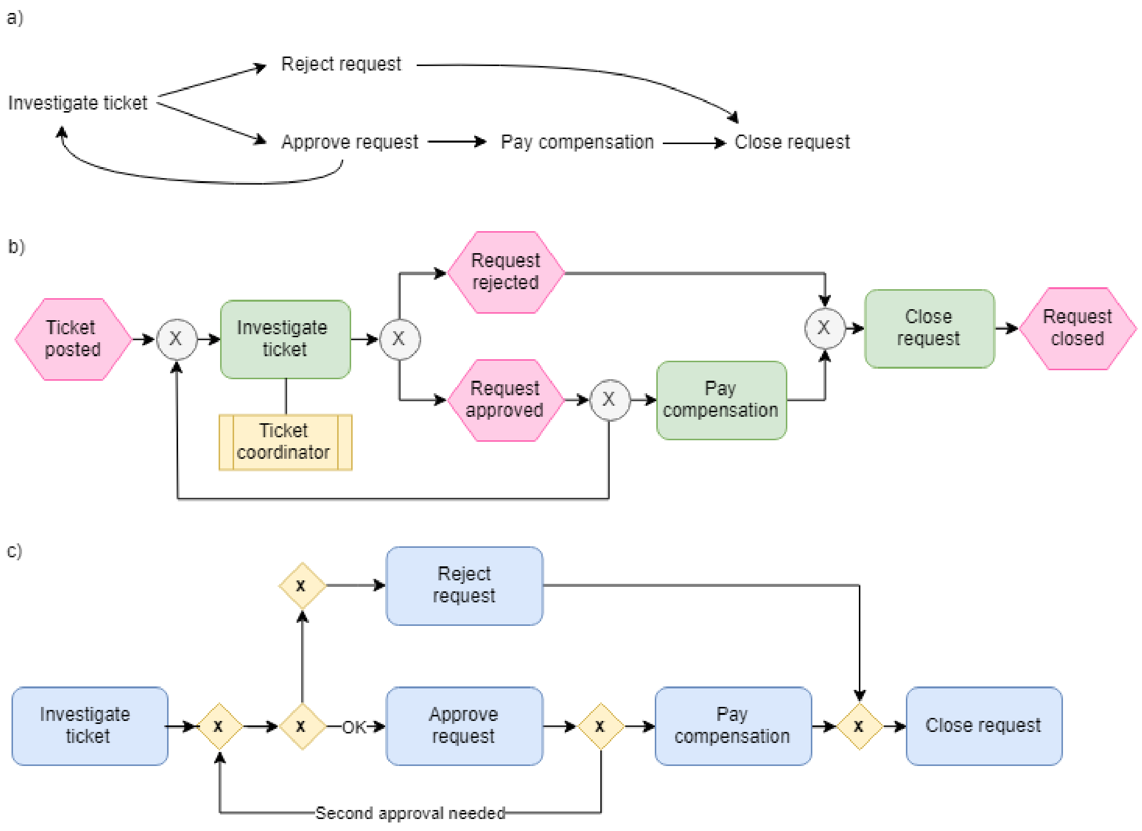


Figure 3.2: Different formality levels in process models. a) Directly-follows graph. b) EPC diagram (operational model). c) BPMN model (executable model).

The lowest level of formality can be found in *fuzzy models* (no example given, as this could be anything). These models do not have any well-defined semantics and hence make it impossible to replay traces on these models. Slightly more formalized, there are *directly-follows models*. These models have activities and unidirectional arrows between those activities that illustrate that certain activities occur after each other. These model do not offer any constructs that make the model easier to understand (i.e. 'syntactic sugar'), which can make the models illegible once they become large and more complex. Next, there are *operational models*. These models offer some visual constructs that enable the reader to understand more complex structures, such as AND-gates and OR-gates, coloring for different entity types, and possibly more. Two good examples of such models are Event-driven Process Chains (EPC's), and Causal Nets (C-nets). Operational models are more formalized, which allow process instances to be explained by the model, but the models cannot be executed automatically to verify such traces. This is because some of the constructs in operational models require human judgment for correct execution (Buijs et al., 2014). Last, there are *executable models* that do allow execution of process instances on top of them. These models have well-defined syntax and do not work if model elements are misplaced such that the model becomes syntactically incorrect. Examples of these types of models are Business Process Modeling Notation models (BPMN models) and Petri nets.

Next to the formality of the process model, it is important to know what the process model describes. A process model can be *descriptive*, meaning it describes what is happening in the process. If this is the case, nonconformities as a result of conformance checking would mean that the model does not accurately describe what is currently happening in the organization. On the other hand, the process could be *normative*, meaning that it describes what is supposed to happen in the process. If nonconformities occur in this case, it means that the process does not conform to the planned process flow. Aside from these main types, process models could also describe an ideal process flow (happy flow), in which case nonconformities would simply indicate that the process does not always have ideal conditions. Knowing what exactly the process model describes is pertinent to a successful conformance checking analysis. In the case of our example, the model is a BPMN model. This is the highest level of semantics (executable), because it is a model that can be executed. As mentioned before in Chapter 1, the model describes the intended process and is hence *normative* of nature.

3.2 Possible Congruity Issues

This section describes what kind of congruity issues commonly occur in event logs and process models in practice. The section starts by describing issues on a high level, after which the resulting data-level issues are covered. The matters discussed in this section do not form an exhaustive list. There may be more issues that occur with different processes, information systems, and models.

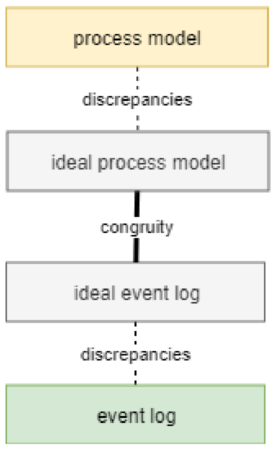


Figure 3.3: Discrepancies relative to the ideal component

There are several different issues that can occur in either *input component* of a log-model pair (event log or process model). Note that these issues are relative to the *ideal component* in the context of the conformance checking analysis, as illustrated in Figure 3.3. An ideal component has congruity with its ideal counterpart. Together, the ideal log-pair combination is considered to be adequate to perform a conformance checking analysis with. The ideal components consider exactly all the process steps that are important for the analysis. This means that if we consider a component to have a particular issue, this is a discrepancy between the actual component and the ideal component. The discrepancies are the aspects that are adapted in the subsequent chapters to develop said ideal component. On a high level, the following are the dimensions of issues that can exist in a component.

- Component may be too granular/too abstract
- Component may contain too many/too few process steps
- Component may have syntactic issues
- Component may have semantic issues

Note that components may have multiple of these issues. Moreover, it is possible that a component is both too granular and too abstract relative to the ideal component at the same time, for different parts of the process. It is important to identify the individual process steps to which this applies, rather than identifying that a component has a 'global' issue (e.g. is too abstract for the entire process). While possibly true, this could potentially not be a correct depiction of the complete situation. Only with issues identified on an individual (i.e. local) process step basis is it possible to cover all congruity issues. The following subsections explain the various potential issues as listed above.

3.2.1 Too granular/too abstract

In Section 3.1, the difference in purpose between an event log and a process model was briefly discussed. If there is a difference in purpose between the two, it is often the case that the two components lie on very different levels of abstraction. Whereas the event log is typically low-level (because it originates from the low level data), the process model manually created for a better business understanding is typically higher level. Although less common, the reverse can also occur in practice. A process model can describe a very detailed working procedure meant for operations, while the event log may only describe the major events in this procedure.

When the abstraction level of process steps in either component is much higher than in the other, it is impossible to make a direct mapping from the event log events to the process model activities. For example, if a process model includes an activity *Process Invoice*, and the event log contains three different events *Scan Invoice*, *Read Invoice*, *Classify Invoice*, conformance checking techniques would make very little sense of this input. Results would indicate that the process execution includes events that are not supposed to occur and that desired process steps are missing. In this example, depending on what is needed in the conformance analysis, it is arguable that this part of the process model is too abstract relative to the ideal situation, which means that it needs to be adapted to be more granular. It may also be the case that the event log is too granular, in which case the event log needs to be elevated to a higher level of abstraction. Using the same example, the ideal components may indeed use the activity *Process Invoice*, rather than the three loose activities.

This congruity issue is quite high-level and propagates to lower-level issues in multiple ways. Other issues can occur because of discrepancies in abstraction level are not unique to this particular problem. These are covered in the following subsections.

3.2.2 Too many/too few process steps

Event logs are created with data retrieved from information systems. It is often the case that the raw data contains much more information than actually necessary, as many small steps may be recorded that are not relevant to the overall execution of the process. Depending on how the event log is generated, this could result in an event log that is filled with unneeded information (noise), which can pollute the results of conformance checking. In a polluted log, the event log contains more process steps than the ideal event log. Process models are typically built to convey the major steps of the process, excluding all the smaller steps for better legibility (see the *simplicity* metric discussed in Chapter 2). Contrariwise, the process model could contains too few steps, when compared to the ideal process model. When using conformance checking techniques on unadapted input with these symptoms, the results will show that many log events cannot be executed on the model, giving the false impression that unintended behavior occurred.

On the other hand, a process model may describe manual (i.e. offline) process steps outside the scope of the information system. Because these events could never be retrieved from the data, the ideal process model does not contain this information. The same situation occurs when the information system landscape makes it impossible (or very difficult) to extract the data for a particular event, for which the events ends up missing in the event log. This results in a scenario where the ideal event log describes more information than the actual event log. With these symptoms, conformance checking results falsely indicate that process instances are not following all the steps in the modeled process.

Note that both of these scenarios can occur at the same time in one log-model pair. Some process steps may be missing in one component, while some others may be extraneous in the same component.

3.2.3 Syntactic issues

Syntactic issues in the context of this work are issues that have to do with the *process structure*. Due to reasons mentioned in 3.1, it could be the case that the process model at hand contains different process structures than the ideal model. A process model that was created for a specific business understanding may not contain the structures needed in the context of the conformance analysis. Relevant structures are starting and end points, choices, parallelism, loops, and the general order of process steps. When applying conformance checking techniques to input that has syntactic differences, false nonconformities may appear. To replay a particular trace with a syntactic difference relative to the model, the replay execution will include log moves (steps taken in just the event log, not the model) to execute process structures that were not originally supported by the model.

For example, assume that the ideal model allows payment of the compensation in two terms. In such a scenario, the *Pay compensation* would occur twice in a row. When the *Pay compensation* activity occurs twice in a row in a trace of the running example, the first *Pay compensation* execution will be a valid move, because it exists in the model. However, the second execution is not supported by the model and will hence show a log move in the replay, falsely indicating a nonconformity. In this scenario, the process model at hand does not have the same syntactic structures as the ideal model and needs to be adapted.

3.2.4 Semantic issues

There can also be issues that are semantic of nature. These have to do with the meaning of model/log elements. A component may contain a(n) (non-syntactic) element that is different than the ideal component, even though they resemble the same thing. Because of these issues as these, conformance checking techniques will indicate that there are problems, although there is really nothing wrong other than the naming of certain activities. These problems can occur because event logs often include technical terminology that does not directly align with the terms used in business-level process models, although they may mean the same things in the context of the process. It is important to have a clear idea of what is needed in the conformance analysis and to align the components such that they use the same terminology for similar events. To this end, the process model and event log have to be adapted towards their ideal versions. Issues that are semantic of nature typically have to be detected with the help of (technical) process expertise.

3.3 Domain Knowledge

Now knowing why the process model and event log often may not have congruity, it is important to identify the congruity issues and repair these in order to perform meaningful conformance checking. Because either of the two inputs needed for traditional conformance checking are not reliable, there is need for a third, more abstract input: *domain knowledge* (see Figure 3.4).

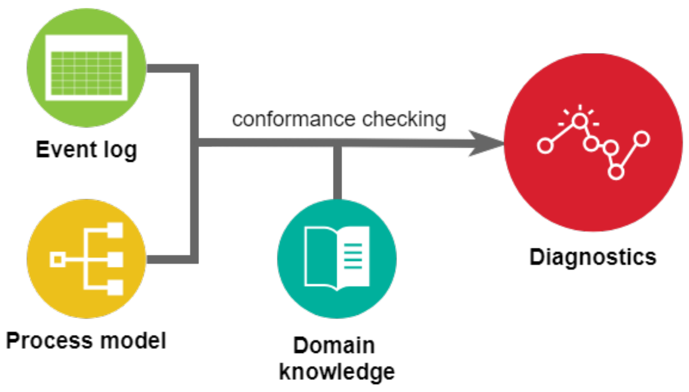


Figure 3.4: Third input for generalized conformance checking: domain knowledge

In this work, domain knowledge refers to an extensive understanding of several separate elements. First, it covers knowledge of the information system landscape from which the event log was extracted. In this, it is especially important to know how the event data flows through the system. Secondly, domain knowledge covers an understanding of the operational process (sometimes referred to as *system* in literature). To generate meaningful insights from a conformance checking analysis, it is critical to know what the process is supposed to achieve and to know how this goal is reached. Third, it covers an understanding of the goal of the conformity assessment. A clear understanding of what is and what is not a nonconformity is essential. This understanding could be structured in the form of a conformance checking specification, with detailed descriptions of the kind of nonconformities that should be retrieved.

Summarized, domain knowledge covers:

- Understanding of the *information system landscape* supporting the process, including the data architecture and the flow of data
- Understanding of the *operational process*, including steps and goal(s)
- Understanding of the *goal of the conformity analysis*, including what is and what is not a nonconformity

Many false nonconformities will appear when performing conformance checking on a log-model pair that does not have congruity. Domain knowledge is required to make a distinction between the nonconformities that are caused by mere incongruity (technical nonconformities), and those that are real nonconformities. Moreover, domain knowledge is required to solve identified congruity issues, as shown in the following chapters.

Chapter 4

Identifying Log-Model Pair Congruity Issues

To identify the congruity issues that need to be solved it is pertinent that the input is ready to be used in appropriate techniques. For issue identification, the event log needs to be replayed on top of the process model. As discussed in Section 3.1.2, only an executable model allows for this type of techniques to be applied. If the process model at hand is not yet on an executable level, the model needs to be translated and elevated to this level for replay techniques to be applied. Model translation is a complex problem and can be very time consuming depending on the abstraction level and quality of the starting model. In the case of the running example, the model is already at an executable level. This makes it easy to create the model in a format that is recognized by most tools available, such as BPMN or Petri net. The translated models of the example can be found in Appendix A. For an event log, the formatting is not as awkward as it is generally simple to transform the event log to a format that is input-ready, be it in a database or as a flat file source. Again, the full event log used can be found in Appendix A.

Using ProM and applying the *Replay a Log on Petri Net for Conformance Analysis* action on the unadapted (content-wise, not format-wise) incongruous input will show nonconformities. In the issue identification stage of the approach, first all the nonconformities need to be identified, after which it is needed to classify all of these nonconformities into one of the following categories.

- *Resolvable incongruity*: this is a false (technical) nonconformity that can be solved.
- *Irresolvable incongruity*: this is a false (technical) nonconformity that cannot be solved.
- *Real nonconformity*: this is an actual nonconformity that is meant to be seen as such.

To perform this classification, nonconformities need to be investigated. During the investigation, domain knowledge is needed to assess the severity of the issue and to classify the nonconformity accordingly. The following will show an identification procedure, using the running example.

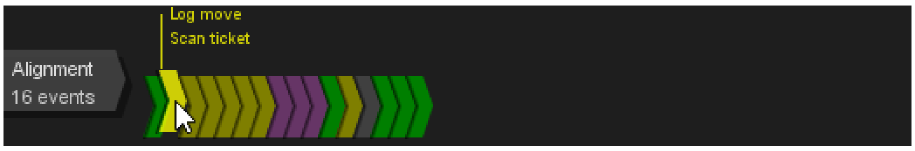


Figure 4.1: Alignment of a case: nonconformity on log event *Scan ticket* highlighted

Figure 4.1 shows an alignment of a process instance (case 6) in the event log on the executable Petri net of the original model. This view can be found in the *Project Alignment to Log* visualization from the results of the *Replay a Log on Petri Net for Conformance Analysis* action.

The alignment shows several different colors. A green color indicates a *synchronous move*. This means that during the replay execution, it was possible to perform an activity in the model at the same time as an event found in the log trace. Hence, this indicates a completely valid step in the process that is aligned over the log and the model. A yellow color indicates a *log move*. This means that during replay, it was impossible to find the process activity in the model that belongs to the current event in the log. To continue the replay execution, the algorithm simply performs the move on the event log and then continues the replay. Indicated in purple are *model moves*. These moves are not found in the event log but have to occur in order to reach completion of the trace. Therefore, the algorithm performs these as model moves in order to continue executing the trace. Lastly, there are moves seen as dark gray. These are *unobservable moves*. The model can include invisible transitions that are needed to provide the right model constructs. These invisible moves are sometimes added to the model when translating it to the right format. Unobservable moves are performed by using these invisible activities in a process model, which can sometimes be needed to bring the replay of a trace to proper completion.

The rest of this chapter discusses the identification and classification of individual congruity issues and nonconformities in Section 4.3. Before diving into individual issues, Section 4.2 goes over the recognition of patterns over several traces to identify the main process areas for incongruity.

4.2 Pattern Recognition

Analyzing traces for incongruity on an individual basis can be a cumbersome and lengthy process. Additionally, certain process elements may miss a proper classification if they are recognized in one trace but not in another. It is therefore important to look at trace patterns that occur often rather than looking at individual trace alignments in the first stages of the issue identification procedure.

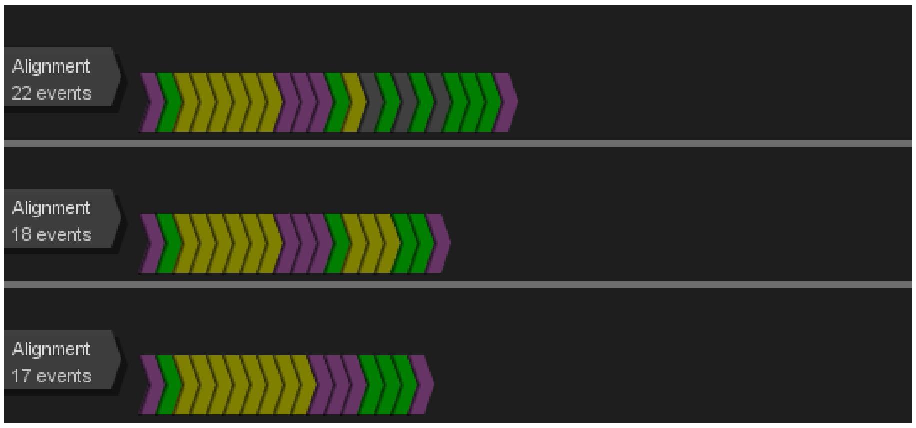


Figure 4.2: Three alignments of the running example

The patterns that the traces create in the alignments will look similar because common process elements in an incongruous log-model pair will cause the same nonconformities. This makes it easier to find the most common congruity issues. In general, it is possible to assume that the majority of the cases in the process have a conforming process execution. Patterns that occur in the majority of the cases are hence likely to be caused by incongruity. Figure 4.2 shows three alignments of the running example that have similar patterns. Before diving into individual trace steps, these patterns should be investigated briefly.

Figure 4.3 highlights the pattern that can be seen at the start of the traces. The pattern exists of a model move move on the start event, followed by a synchronous move on the 'Register request'

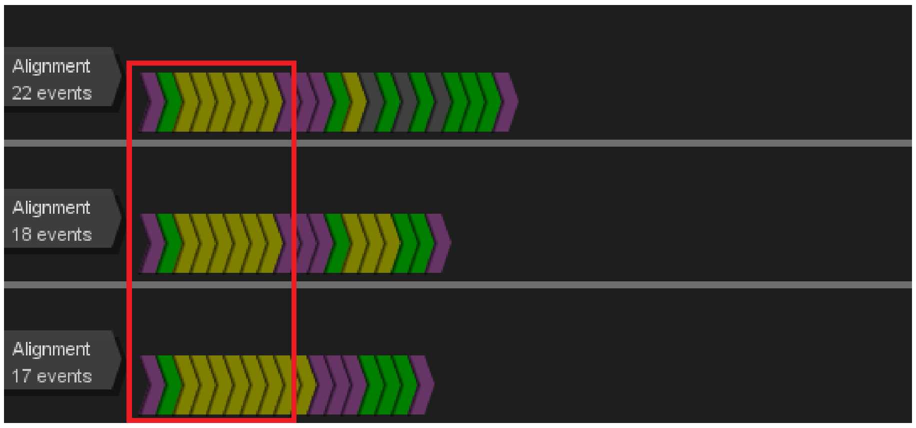


Figure 4.3: Pattern identified at start of running example

step. After that, some log moves are performed on the various OCR events of the log, after which the pattern ends with the 'Post ticket' step. This pattern shows that there is an issue with the OCR process steps, which can then be analyzed in further detail. Figure 4.4 shows another pattern that can easily be seen in the alignments. This pattern starts with the 'Post ticket' log move seen previously, followed by three model moves on the activities 'Review ticket', 'Register ticket', and 'Investigate ticket'. Finally, the pattern ends with a synchronous move on the 'Approve request step'. All activities seem to be valid process steps that do not appear in the event log. It is therefore necessary to look into this problem in more detail. The next step would be to look at these patterns in more detail by picking out individual traces and analyzing the causes for the nonconformities in these traces. This next step is covered in the following section.

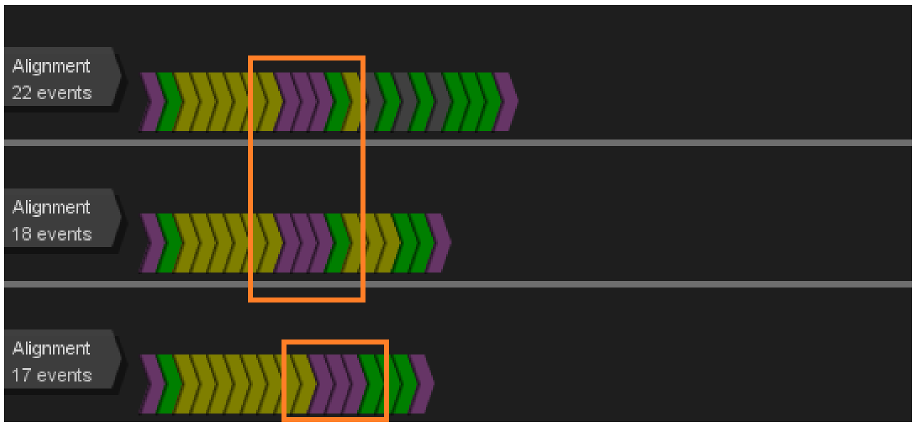


Figure 4.4: Second pattern identified in the running example

4.3 Issue Identification and Classification

This section covers the issue identification and classification process on individual traces. The first alignment discussed in Figure 4.1 starts with a synchronous (green) move. The *Register request* event in the trace maps correctly to the first activity in the process model. The first nonconformity seen shows a log move on the event *Scan ticket*. This implies that this log event does not have a mapping to a corresponding activity in the process model. When looking at the process model, it is immediately obvious that there is only one activity possible after the initial *Register request* activity, namely *Review ticket* (see Figure 4.5). Domain knowledge then needs to be applied to classify this nonconformity accordingly.

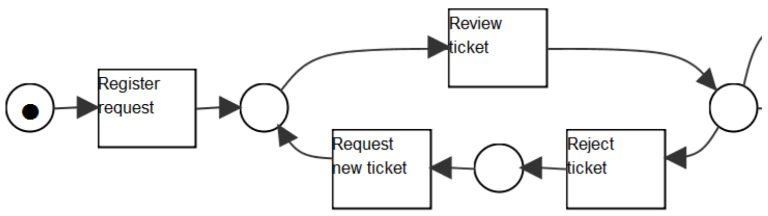


Figure 4.5: Starting activities in the model

Nonconformity identified: Scan ticket occurs in event log but does not exist in process model

To classify the nonconformity correctly, the first question that should be asked is: *Is the nonconformity found a real nonconformity?* Should the answer be 'Yes', then the classification is simple. If the answer is 'No', then it is needed to dig deeper to see whether the nonconformity (or simply incongruity at this point) is resolvable or not. This step is more complex, because the resolvability of the issue depends on multiple—possibly interrelated—factors. Domain knowledge needs to be applied to make a correct assessment of the resolvability of the incongruity.

Although there could be more factors that play a role in the resolvability depending on the incongruity at hand, the most important general factors are *1) the importance of the relevant data*, *2) the complexity of the issue* and *3) the (availability of) additional relevant data*. The importance of the data relevant to the issue is a factor, because resolving the issue could be as simple as deleting the data causing the issue. If the data causing the issue is not of any particular importance, then deletion is the simplest option to resolve the nonconformity. Second, the complexity of the incongruity is evidently important. Assessing the complexity gives a good indication of whether the issue can be solved. In some cases, solving an incongruity is possible but would require a huge amount of effort, for instance because the data needed is hard to retrieve from the system or because the process model requires a very complex structure to solve it. Thirdly, the other relevant data plays a role because it is possible the issue does not revolve around one single process step, but has to do with multiple steps. In this case, it is needed to look at all the relevant steps to make an appropriate classification. Additionally, it may be that there is need for extra data from the information system or another source, which is not always available. As can be seen, these factors require domain knowledge to be applied in order for them to be assessed effectively. Based on these factors, it should be possible to make an assessment on the resolvability of the incongruity. A decision tree for classification based on these criteria is presented in figure 4.6.

Looking at the previously identified nonconformity in the example, let us assess the factors individually to find the correct classification. The importance of the *Scan ticket* event in the log is debatable. From one perspective, you could argue that this is an important step in the process, because without it, none of the other steps could take place. On the other hand, scanning a ticket is a mandatory step which does not involve much risk in terms of conformity. Based on the context

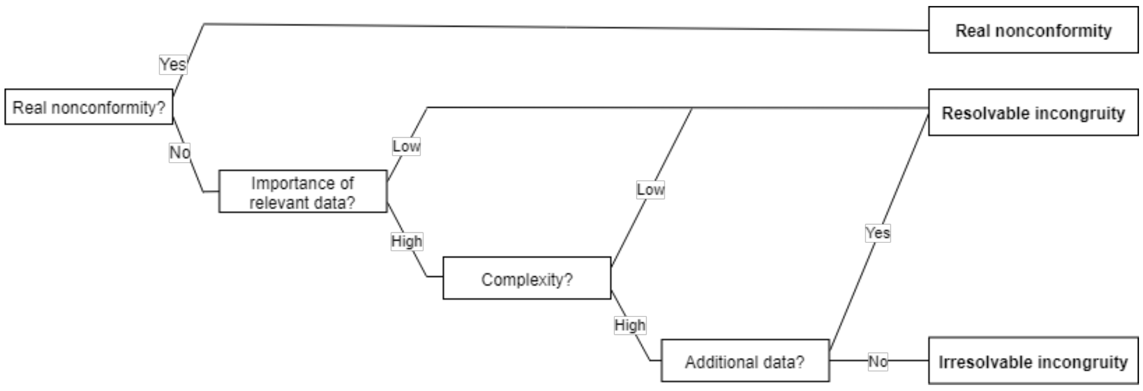


Figure 4.6: Classification decision tree

of the conformance analysis taking place, you could say that the scanning activity is simply part of the *Register request* activity found in the process model. For this reason, we decide that the importance of the event is low. According to the decision tree in figure 4.6 we can already classify the nonconformity as resolvable solely based on this information.

Nonconformity identified: *Scan ticket* occurs in event log but does not exist in process model
Classified as: Resolvable incongruity

Let us cover another example. In a different case, we find a log move on the *Cancel approval* event in the trace (see figure 4.7). It occurs right after the first approval in this trace, which is a natural place to find such an event (using domain knowledge). Nevertheless, the event does not yield a synchronized move between the event log and the model. Looking at the process model (figure 4.5), we see that the the *Cancel approval* activity is not modeled.

Nonconformity identified: *Cancel approval* occurs in event log but does not exist in process model

Following the decision tree, it is first necessary to assess whether this is a real nonconformity or not. This step is not a nonconformity, because this step can be critical in a process execution. Hence, we consider it to be an incongruity. Then, the importance of the event is assessed. Cancelling an approval is a step in the process that is needed, therefore we can consider the importance *high*. This means that we cannot simply get rid of this information. The complexity of this congruity issue is quite low. This issue could be solved by introducing the *Cancel approval* activity in the process model. The resulting classification is a *resolvable incongruity.*

Nonconformity identified: *Cancel approval* occurs in event log but does not exist in process model
Classified as: Resolvable incongruity

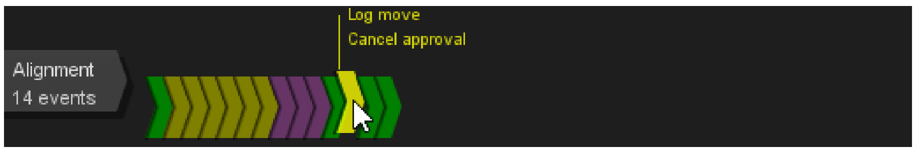


Figure 4.7: Alignment of case 10 - Log move on *Cancel approval* highlighted

In some cases, the congruity issues can cause other issues to appear around them. In the example

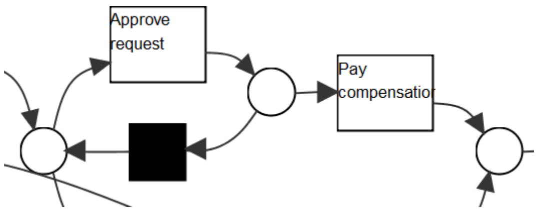


Figure 4.8: Model excerpt with request approval and compensation payment

above, the *Cancel approval* activity was missing in the model. Looking at the prior and subsequent events in the trace, it can be seen that the *Pay compensation* event occurred, after the approval cancellation, as can be seen in figure 4.9. Evidently, payment of a compensation of which the approval has been revoked is not desired behavior in the context of the business process. Hence, this is another nonconformity that needs to be classified as such. The alignment shows this as a synchronous move because the model does not recognize the *Cancel approval* activity, but does recognize the *Pay compensation* activity. The synchronous move on behavior that is undesired is a false positive, and hence a nonconformity. Note that this nonconformity would have appeared as an invalid move if the approach of identifying and solving incongruity issues would have been performed in an iterative way. After solving the cancellation of approvals, the invalid payment of the compensation would have been identified.

Nonconformity identified: *Pay compensation* occurs in event log after the *Cancel approval* activity
Classified as: Real nonconformity

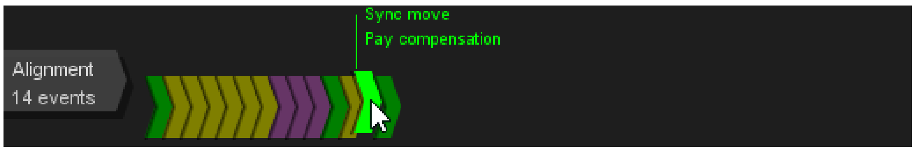


Figure 4.9: Alignment of case 10 - Sync move on *Pay compensation* highlighted

When the most occurring nonconformities have been identified and appropriately classified, it is possible to start the process of repairing the process model and event log to remove discrepancies between the components and their ideal versions. This is an iterative process, which means that after initial reparations, the identification process is to be done again to find any new incongruities.

Chapter 5

Adaptation Process

After identification and classification of nonconformities into the three distinct categories (resolvable incongruity, irresolvable incongruity, and real nonconformity), it is possible to start the process of adapting (improving) the event log and the process model to solve any discrepancies. Doing this should create a log-model pair that has congruity in the context of the analysis to be performed, allowing current conformance checking techniques to generate meaningful results with regards to process nonconformance. The high-level method can be seen in figure 5.1. The rest of section 5.1 covers the approach more in-depth and summarizes the approach as a whole. Then, Section 5.2 describes how adaptation of the event log and process model should be done. At the end of the chapter, Section 5.3 gives an overview of the systematic approach.

5.1 Systematic Approach

This section will go over the high-level approach on how to overcome congruity issues in an event log and a process model. Figure 5.1 below shows the diagram seen before in Figure 3.4, with the addition of the *adaptation of the model and event log.*

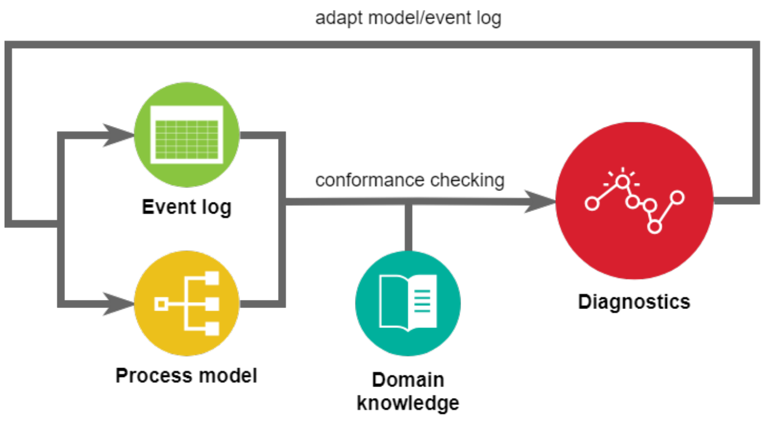


Figure 5.1: Iterative approach to conformance checking on incongruous input

As described by Figure 5.1, this conformance checking approach is iterative. Reasoning for this is twofold. First, applying conformance checking techniques on all incongruous input data yields a big number of false nonconformities. To keep the number of nonconformities manageable, it is suggested to perform the issue identification method on a random subset of the entire event log, of which the size depends on the original input and the number of possible nonconformities. It is wise to start the iterative process with a smaller subset of data in case there are many nonconformities, for the reason that the results are more manageable. Some nonconformities require some in-depth investigation, after which it can be beneficial to adapt the process model and event log quickly. Hence, a smaller number of nonconformities in the subset allows for faster iterations. This is important for the second reason, as the iterations are very beneficial in resolving all incongruities. Solving one incongruity by adapting the event log or the process model can introduce new incongruities further along the process. These newly introduced incongruities could be identified in one iteration by doing manual inspections, but that is more cumbersome due to having to inspect seemingly valid process executions to identify those nonconformities. For these reasons, we use an iterative approach in which each iteration uses a sample of the complete dataset as input.

Looking at a single iteration, the approach works as follows. The event log and process model are used as input for a conformance checking technique (replay techniques are applied in this work). Also applying domain knowledge allows the analyst to find relevant issues (= nonconformities) in the conformance checking results (**Diagnostics** in Figure 5.1). Because the event log and the process model do not have congruity, many nonconformities arise in these diagnostics. However, many of these nonconformities are not actual cases of process nonconformance, but rather nonconformities that stem from the incongruity of the input as seen in Chapter 4. Therefore, it is needed to look at the nonconformities in detail and classify these into the three distinct categories: *resolvable incongruity*, *resolvable incongruity*, and *real nonconformity*. For each *resolvable incongruity*, domain knowledge should help in determining how to tackle the problem and make the input more congruous. Section 5.2 will cover the adaptation step. Table 5.1 can guide as a reference for the possibilities to solve the issue at hand. Next, there are incongruities that cannot be resolved. When incongruities arise that are too complex to resolve, these incongruities can be classified as *irresolvable*. In this case, we acknowledge the fact that what seems to be a nonconformity in the conformance checking results is caused by technicalities rather than actual process nonconformance. For these nonconformities, it makes sense to label them as an incongruity if the analysis tooling used allows for this. In this work, Celonis is used to *whitelist* this type of nonconformities. By doing this, the incongruity is no longer shown as a process nonconformance, even though this still appears in the log-model combination. Then, it is possible that a nonconformity in fact shows a true process nonconformance that we wish to identify. These are classified as *real nonconformities*. It is important to classify these, as it is important to keep retrieving these nonconformities as such in every iteration. These nonconformities are to be *blacklisted*. This is not an option in any currently available tooling, but making a note of the blacklisted nonconformities confirms that they are present in the input and that they should be continually retrieved as a nonconformity, also (especially) when the log-model pair has congruity. When classification is complete, the resolvable incongruities are tackled by adapting the process model and the complete event log (rather than just the subset used as input in the iteration), such that the incongruities found in the current iteration are eliminated. The result is a new pair of process model and event log that can be used in the next iteration.

After performing several iterations with sampled input, the number of unclassified nonconformities should drop significantly. Iterations can be performed until no more incongruities are retrieved from the conformance checking techniques, or until resolving new incongruities would cost more effort than desired. In the last case, remaining resolvable incongruities should be reclassified to *irresolvable*, after which they are whitelisted at the end. The result of this approach is a log-model pair that has much more congruity than at the start, making them suitable for contemporary conformance checking techniques.

The adaptation of the log and model can be done in various ways. The next section will discuss this problem in more detail.

5.2 Adapting the Log and Model

This section describes the adaptation of the event log and the process model to overcome the identified *resolvable incongruities*. First the adaptation of the event log is covered. The adaptation of the process model is discussed thereafter. Note that congruity issues sometimes revolve around both input components (i.e. the event log *and* the process model. In these cases, it is beneficial to adapt both components at the same time to make sure that the issue is solved on all accounts.

5.2.1 Event log

First, the issues that require event log adaptation are briefly described, after which some improvements will be performed on the event log from the running example to provide a more tangible feeling of the necessary steps. Adapting the event log can be done in any tooling that allows adaptation of textual fields. In this research, adaptations are made by means of SQL statements, as the event log used resides in a local SQL database.

Too granular/too abstract
If the event log is too granular, it means that it has too much detail included compared to its ideal counterpart. This problem can be solved by simply removing events from the log that are not relevant for the conformance analysis. When the detailed events have been removed, it may be needed to change the name of a particular event to become more abstract (i.e. high-level), better suiting the analysis.
If the log is too abstract, the opposite should be done. More detailed events should be introduced to make the log more detailed. Not that this is only possible if the information system used records more granular information regarding the process. Again, it may be needed to rename some activities to describe the events more accurately.

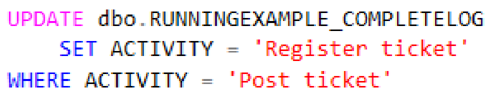
Too many process steps/too few process steps
If the event log has too many or too few process steps, this is a matter of removing or adding events. Removing is not a problem, since the event already exists in the log and it is a matter of deleting all the occurrences of this event from the log. The addition is sometimes impossible when the information system cannot provide the needed information. In this case, nonconformities that appear because of this problem can be considered irresolvable (and hence whitelisted).

Syntactic issues
Syntactic issues in the event log occur when the structure is not correct. This is an issue that is hard to identify in event logs, because they are built on empirical data in which the structure is true to reality. Nevertheless, some information systems do not store all the information required to make a unique structure in the event log. Timestamps can sometimes be limited to just the date rather than data *and* time of the event occurring. It is important to be able to distinguish which activity occurred first if the data does not directly provide this information. This can be done by restructuring the event log and adding a *sorting* column that defines a unique sequence of events.

Semantic issues
Semantic issues simply have to do with the meaning of particular activities. It the running example, we see that many activities in the event log have technical names. These names often do not align with the ideal version of the event log, in which terms are more business-oriented. In such cases, the names of events need to be adapted.

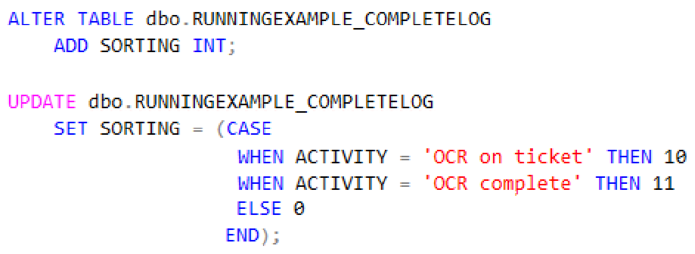
Adaptation
As mentioned, event log issues are tackled by using SQL queries on the database. Using the running example, what follows are a few adaptations that were made to the event log. Figure

5.2 shows an example SQL query that is used to make adaptations to the semantic issues in the event log. This query changes the name of all the events of a particular type in the complete event log. The example shown changes the events with activity name 'Post ticket' to 'Register ticket', which is the term used in the event log. After performing this change, the event log and the process model will match on this activity. Figure 5.3 shows another example used on the running example. In this case, the issue is a syntactic issue in which the two events 'OCR on ticket' and 'OCR complete' do not define a unique order due to these events having the exact same timestamp, as seen in Table 1.1. The script first adds a new column SORTING to the event log, after which an order is defined by giving a sorting number to the activity. With this, it is defined that the event 'OCR complete' always occurs after 'OCR on ticket'.



```
UPDATE dbo.RUNNINGEXAMPLE_COMPLETELOG
    SET ACTIVITY = 'Register ticket'
WHERE ACTIVITY = 'Post ticket'
```

Figure 5.2: SQL query for adaptation of the event log for a semantic issue



```
ALTER TABLE dbo.RUNNINGEXAMPLE_COMPLETELOG
    ADD SORTING INT;

UPDATE dbo.RUNNINGEXAMPLE_COMPLETELOG
    SET SORTING = (CASE
                    WHEN ACTIVITY = 'OCR on ticket' THEN 10
                    WHEN ACTIVITY = 'OCR complete' THEN 11
                    ELSE 0
                   END);
```

Figure 5.3: SQL query for adaptation of the event log for a syntactic issue

5.2.2 Process model

For the process model, the same issues exist that were previously discussed in Chapter 3. First, the various issues and their possible solutions will be briefly described. Then, modifications are made to the process model from the example to showcase how the solutions can be applied. The process model can be adapted in any applicable tooling. In this work the input is translated to a BPMN model which allows us to make modifications in any BPMN modeling tool.

Too granular/too abstract
Similar to the event log, it can be the case that a process model is too granular or too abstract relative to the ideal model. If it is too granular, this means it includes too much detail and that it should be brought to a higher level of abstraction. In some cases, this can be done by removing activities from the process model. Some other cases may require grouping of several lower-level activities to form one higher-level activity. In this case, this is a restructuring solution. If the model is too abstract, activity addition can be a solution. When adding activities on a more granular level, this may require some changes to the process structure, as subprocesses may be introduced. Renaming activities may be a necessity in both cases, as a change in the abstraction level may require a change in the activity name to describe the process step accurately.

Too many process steps/too few process steps
Having too many or too few process steps in a process model is an issue that is simple to solve with the addition or removal of an activity. The analyst is not limited to the data available in a process model like they may be for the event log. Therefore, incongruity issues of this nature simply require the addition or removal of relevant activities in the process model.

Syntactic issues
Syntactic issues in the process model occur when the process flow does not have the correct structure. In particular, this may happen when there is a deviation from the linear process flow, introduced by process constructs such as choices and parallelism. If the process model lacks such structures, it is needed to introduce these to the model. Adding these constructs may require the removal and addition of other activities as well. It could also be that the model includes process structures that are not represented in the event log. In that case, these constructs should be removed from the model instead, possibly along with the relevant activities.

Semantic issues
Similar to this semantic issues in the event log, semantic issues in the process model concern the meaning of activities. It could be the case that the same process steps are present in the event log, where they are named differently. Solving these semantic issues require adaptation of the naming of activities in the process model.

Adaptation
Adaptations of the process model are performed by making changes to the model in their respective format. Figure 5.4 shows an example of a model adaptation in the context of the running example. The change is the addition of an activity 'OCR on ticket complete' that introduces the OCR steps of the event log to the model. The activity is given a different name however, as the event log was considered too granular.

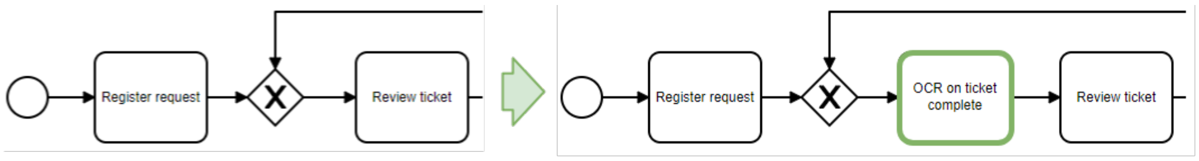


Figure 5.4: Adaptation of the event log for an abstraction issue: grouped activity added

5.2.3 Approaches

Table 5.1 gives an overview of all the approaches mentioned in the previous sections. This table can be used as reference when adapting a process model and event log to resolve their incongruity.

Table 5.1: Possible approaches for resolving congruity issues

		Adaptation Approach			
		Addition	Removal	Restructuring	Renaming
	Incongruity				
Event log	Too granular		x		x
	Too abstract	x*			x
	Too many process steps		x		
	Too few process steps	x*			
	Syntactic issues	x*	x	x	
	Semantic issues				x
Process model	Too granular		x	x	x
	Too abstract	x		x	x
	Too many process steps		x		
	Too few process steps	x			
	Syntactic issues	x	x	x	
	Semantic issues				x

5.3 Step-by-step Approach

The previous sections of this work described the systematic approach in detail. This section provides a summary of the elaborated approach in a step-by-step guide. Using this guide would allow an analyst to perform conformance checking on any pair of event log and process model that is incongruous.

Input: process model, event log, domain knowledge
Result: conformance checking diagnostics

1. Take a sample of the event log
2. Perform initial conformance checking (replay) on the event log sample and the process model
3. Identify main nonconformities
4. Classify these nonconformities as:
    - *Resolvable incongruity* (to resolve)
        - try to indicate underlying issue using Section 3.2.
        - try to indicate possible solution using approaches matrix in Figure 5.1
    - *Irresolvable incongruity* (to whitelist)
    - *Real process nonconformity* (to blacklist)
5. Address resolvable incongruities on the **complete** input (not the sample) in following order
    - Remove events (noise) from the event log
    - Remove irrelevant activities from the process model
    - Add missing events to log (if possible)
    - Add missing activities to model
    - Address syntax (event grouping/model structure)
    - Address semantics (event/activity naming)
6. Repeat steps 1 through 5 until *no more resolvable incongruities are found* or until *effort > gain*
7. Repeat steps 2 through 5 with complete input
8. If tooling allows, whitelist the remaining incongruities (resolvable and irresolvable)

Chapter 6

Case Study

This chapter describes how the introduced systematic approach was applied in practice on process data retrieved from the ERP system of Wessanen. This conformance checking analysis was performed on the Purchase-to-Pay (P2P hereafter) process of the organization over all their countries of operations in Europe.

6.1 Business and Data Understanding

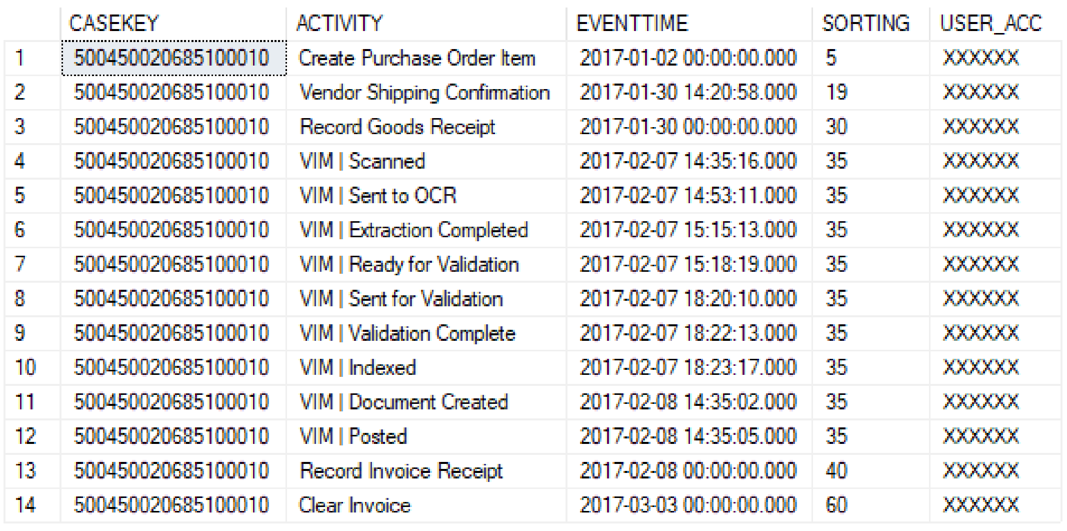
Any conformance checking analysis requires a good understanding of the business process and the data that is recorded during the process. At Wessanen, the P2P process is part of a bigger process called Source-to-Pay, which is the entire procurement process. As the scope for this work, only the P2P part of this end-to-end process was used, as this is the subset of the process that is data-heavy. To achieve a good understanding of all the data available in the ERP system, first the P2P process had to be understood. By speaking to relevant stakeholders and subject matter experts (SMEs), it was possible to get a basic understanding of the process quite quickly.

After a basic understanding of the business, the process, and the data available was gained, it was necessary to dig into the SAP data needed to conduct the analysis. Leveraging the knowledge of SAP SMEs, the right SAP tables were extracted by means of a third-party tool called CSI Data Xtractor. Diving into this data, many questions came up. Some of these questions were more technical and could be solved with some web searching, whereas others required additional business knowledge. This process took several iterations, based on the complexity of the data. Once a more thorough data understanding was achieved, the next phase of the CRISP-DM cycle could begin.

6.2 Data Preparation

6.2.1 Event log

The event log that was used as input for the conformance checking analysis was created from data available in the ERP system of Wessanen. To generate this event log, the artifact-centric log extraction approach was used, developed by Lu (2013). A better understanding of the SAP data was gained with the help of Piessens (2011). Although this event log was created by the authors, it was not made specifically for conformance checking purposes. Instead, the log was extracted for process discovery techniques, applied to gain insights into the actual process flows derived from empirical ERP data. This means that the event log used as input is not an ideal event log in the context of conformance checking. However, the important elements in the context of the analysis are present in the log, which means that the log is not too far away from the ideal event log in this analysis. Using the event log maturity model covered in Section 3.1.1 and fully shown in Appendix B, it is possible to deduce that the event log aligns to level $\star\star\star$. Part of this event log is shown in Figure 6.1.



	CASEKEY	ACTIVITY	EVENTTIME	SORTING	USER_ACC
1	500450020685100010	Create Purchase Order Item	2017-01-02 00:00:00.000	5	XXXXXX
2	500450020685100010	Vendor Shipping Confirmation	2017-01-30 14:20:58.000	19	XXXXXX
3	500450020685100010	Record Goods Receipt	2017-01-30 00:00:00.000	30	XXXXXX
4	500450020685100010	VIM | Scanned	2017-02-07 14:35:16.000	35	XXXXXX
5	500450020685100010	VIM | Sent to OCR	2017-02-07 14:53:11.000	35	XXXXXX
6	500450020685100010	VIM | Extraction Completed	2017-02-07 15:15:13.000	35	XXXXXX
7	500450020685100010	VIM | Ready for Validation	2017-02-07 15:18:19.000	35	XXXXXX
8	500450020685100010	VIM | Sent for Validation	2017-02-07 18:20:10.000	35	XXXXXX
9	500450020685100010	VIM | Validation Complete	2017-02-07 18:22:13.000	35	XXXXXX
10	500450020685100010	VIM | Indexed	2017-02-07 18:23:17.000	35	XXXXXX
11	500450020685100010	VIM | Document Created	2017-02-08 14:35:02.000	35	XXXXXX
12	500450020685100010	VIM | Posted	2017-02-08 14:35:05.000	35	XXXXXX
13	500450020685100010	Record Invoice Receipt	2017-02-08 00:00:00.000	40	XXXXXX
14	500450020685100010	Clear Invoice	2017-03-03 00:00:00.000	60	XXXXXX

Figure 6.1: Excerpt of P2P event log created from Wessanen ERP data

The maturity model states that logs at level $\star\star\star$ are trustworthy, but not necessarily complete. This is the case in the log at hand: all information in the log is gathered manually from SAP tables, but it does not necessarily cover all the steps in the P2P process. As seen in figure 6.1, the event log contains some activities that are very technical. The activities that have the prefix 'VIM' are process steps performed by the Vendor Invoice Management addition to SAP. While some of these steps may be relevant, some are too technical to have any real importance in a conformance checking analysis. The exact congruity issues occurring in this event log are identified in Section 6.3.

6.2.2 Process Model

Process models were created by Wessanen process stakeholders to develop a thorough understanding of the as-is Source-to-Pay processes, as part of an ongoing initiative to standardize and improve their processes. These models were developed by means of workshops, in which the relevant stakeholders could indicate how particular parts of the process are executed in practice. Hence, the models developed based on this information can be considered *descriptive*. Although present, the models are not directly ready for use in a conformance checking analysis. First, the scope of the models is too large. The models cover the entire source-to-pay process, which includes many steps that take place outside of any information systems such as manual steps and phone calls. The scope of these models needs to be brought down to cover the right aspects of the process that are relevant for the conformance checking analysis. Second, the models are not at an executable level, which means that replay techniques cannot be performed on them. Rather, the models are at an operational level in the form of EPC diagrams (discussed in 3.1.2) meaning they can explain the process.

For the congruity issue identification process, the models need to be translated to an executable format such as BPMN or Petri net. Translating a process model can be quite a difficult process depending on the structures involved. For conformance checking, the translation process is quite subjective to the context of the analysis, as obvious incongruities could be tackled right away

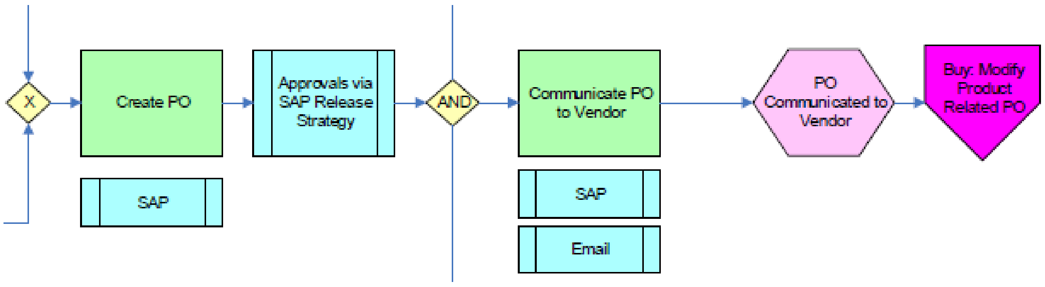

X
Create PO
SAP
Approvals via SAP Release Strategy
AND
Communicate PO to Vendor
SAP
Email
PO Communicated to Vendor
Buy: Modify Product Related PO

Figure 6.2: Excerpt of P2P EPC model made available by Wessanen

during translation. However, for the sake of comprehensibility of this case study, the translation performed on Wessanen P2P models does not include any major adaptations of the process that improve congruity. The fully translated model (into BPMN) can be found in Appendix C.

6.3 Congruity Issue Identification

When the input is translated to the correct formatting, it is ready to be put into ProM for replay, as mentioned in Section 4.1. Because the the process model and event log lie so far apart, the initial alignment is expected to give poor results. Figure 6.3 shows this initial alignment. In this figure, it can be seen that there are no synchronous (green) moves made between the process model and the event log. First, all the events in the log are played (log moves in yellow), after which all the moves in the model are made (model moves in purple and invisible moves in gray). Naturally, because the log cannot be replayed on the model at all, the fitness for this particular trace is 0.

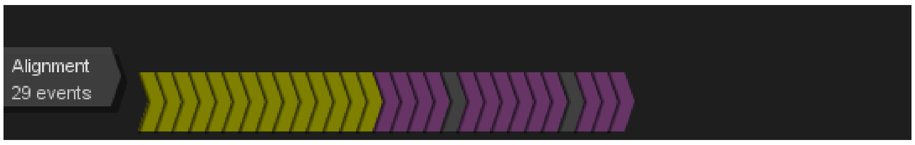


Figure 6.3: Initial alignment input log and model, case 500450022286600030

Looking at this case in detail and going through the different events and using domain knowledge, we see that this trace does not involve any undesired behavior. The trace for this case is the following:

<*Create Purchase Order Item, Vendor Shipping Confirmation, Record Goods Receipt, VIM — Scanned, VIM — Sent to OCR, VIM — Extraction Completed, VIM — Ready for Validation, VIM — Sent for Validation, VIM — Validation Complete, VIM — Indexed, VIM — Document Created, VIM — Posted, Record Invoice Receipt, Clear Invoice*>

Domain knowledge tells us that this case is in conformance with the intended process. Therefore, the alignment should have a high fitness level for the trace not to be recognized as a trace that is nonconforming. Now, let us get into the identification of congruity issues. As covered in Chapter 3, issues are identified by looking at multiple alignments and seeing whether there are nonconforming patterns that occur frequently.
Figure 6.4 shows that the first log move performed in the replay covers the event *Create Purchase Order Item*. The first model move performed is *start_event_null* (not shown in the figure), which is simply the starting point of the model. The second model move performed is *Create PO*. This model move maps to the first log move performed. However, there is no synchronous move possible due different naming (semantic issue) of the same process step. This particular issue occurs in all the alignments of this sampled input, and is hence a incongruity. Seeing how this is due to the

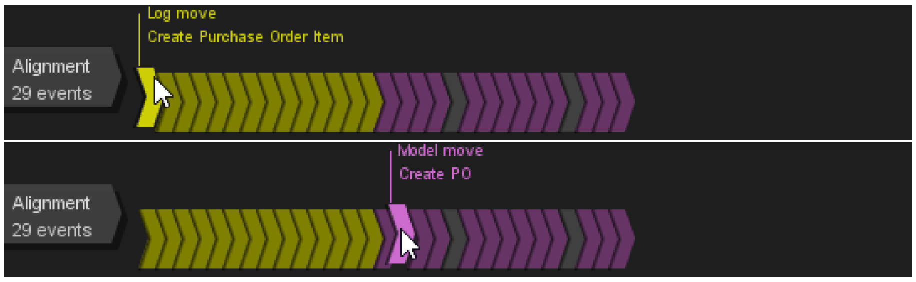


Figure 6.4: Semantic nonconformity identification, case 5004500222866000030

naming of activities, we can consider this to be resolvable.

Nonconformity identified: *Create Purchase Order Item* in log — *Create PO* in model
Classified as: Resolvable incongruity (semantic issue)

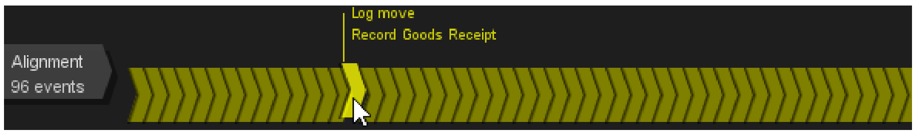


Figure 6.5: Syntactic nonconformity identification, case 5004500221555000010

Next, figure 6.5 shows another nonconformity represented as a log move. In this particular case, the trace includes the sequence <.. , *Record Goods Receipt, Record Goods Receipt, Record Goods Receipt, Record Goods Receipt, ..*>. Again, this issue can be found more frequently when observing other logs. Hence, this can be considered an incongruity. When looking at just the model this seems to be undesired behavior. Domain knowledge however tells us that this behavior is indeed tolerated and quite common in some process instances. For example, in purchase orders for which multiple line items are received separately, or for bulk orders that arrive in several installments. Therefore, the event log more closely represents the desired process and the model should be adapted to support this behavior.

Nonconformity identified: *Record Goods Receipt* can occur multiple times
Classified as: Resolvable incongruity (syntactic issue)

Another issue that is seen in multiple occasions in these alignments is the abundance of events in the event log that provide very little information to the overall process, creating many unnecessary log moves in the replays. In particular, the log contains many events having to do with the Vendor Invoice Management system (mentioned in Section 6.2.1). Some of the steps that happen in this sub-process can be considered noise when we apply domain knowledge. Examples of these events that introduce noise are: *VIM — Ready for Validation* and *VIM — Extraction Completed.* Similar to what was seen in the running example, this problem can be classified as resolvable, simply due to the fact that the information in these activities can be considered obsolete.

Nonconformity identified: *Some VIM events* introduce noise to the event log
Classified as: Resolvable incongruity (too many process steps)

Other type of nonconformities: work on next section to identify

Because this is an actual case study with many nonconformities, several iterations have to be done in order to find and solve all nonconformities. The identified issues above only represent those found in the very first iteration (with unadapated input). Section 6.5 discusses the results of applying the systematic approach, in which an overview is given of the congruity issues that were resolved in every iteration.

6.4 Adapting the Model-Log Pair

For all nonconformities identified, those that are classified as resolvable will be solved by adapting the process model or the event log, depending on the specific problem at hand. While some are very simple to resolve, others may require several iterations of this process for them to be fully resolved. This section of the case study will showcase several adaptations to the process model and event log, as well as the direct results of these adaptations to the replay alignments.

In the previous section, a nonconformity was discussed regarding the naming of event *Create PO*. This nonconformity can be solved quite easily in multiple ways. First, it is needed to assess whether this information is meant to be kept in the analysis. Because the creation of a purchase order is a crucial step in a P2P process, this information needs to be kept. This means we cannot simply delete the information, but the process model or event log (or both) needs to be adapted. Referencing the approaches table, the solution to this problem is renaming this process step in either input component. In this case we choose to adapt the event log, as the process model has a simpler notation that is clear to the reader. To adapt the event log, the following SQL query is executed on the database environment that holds the event log table.

```
UPDATE [DB_NAME].[dbo].[LOG] SET ACTIVITY = 'Create PO' WHERE ACTIVITY = 'Create Purchase Order Item'
```

As mentioned, the query above was used to adapt the event log. It changes all events with activity name *Create Purchase Order Item* to *Create PO*. This solves the identified nonconformity because the model and the log now have the same notation for the event. This means that a mapping can now be made, allowing for a synchronous move in the replay alignment.

The next resolvable incongruity discussed in section 6.3 is the one regarding sequential executions of the *Record Goods Receipt* activity. This issue arises due to a missing structural element in the model. When looking at the model we see that there is a linear execution after the *Record Goods Receipt* activity, with no possibility to go back to this activity after its initial execution. Hence, this possibility has to be introduced by means of a model adaptation as can be seen in the approaches table (table 5.1).

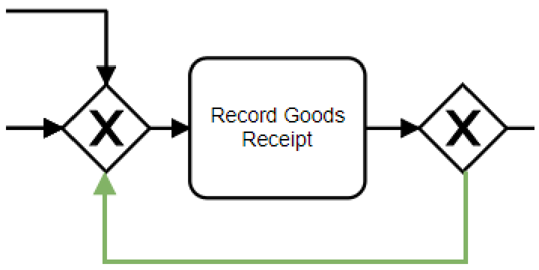


Figure 6.6: Model adaptation required

Figure 6.6 shows the model adaptation required to support the intended behavior. In this case, the addition of an OR-gate and an edge back to the OR-gate preceding the activity is sufficient. This is the case because if this activity is to be performed multiple times, this has to occur in a direct sequential manner.

Section 6.3 also covered the identification of noise in the event log. This is a problem on the event log side, as we do not want to incorporate the activities that create noise to the model. The type of problem is that there are too many activities. The approaches table indicates that the logical solution to this problem is to simply remove the process steps from the component. Seeing how there are several events prefixed with *'VIM —'* that we wish to exclude from the analysis, we delete them with the following SQL query.

```
DELETE * FROM [DB_NAME].[dbo].[LOG] WHERE ACTIVITY = 'VIM | Ready for Validation'
```

The conformance to the model has now improved, because the events in the log that had no mapping to an activity in the model are now removed. Hence, this reduces the number of log moves needed in the replay.

6.5 Results

After applying the modifications in the first iteration it is possible to perform another replay technique on the updated event log and process model. Doing this gives the alignment on the process instance discussed above, shown in Figure 6.7. Although still nowhere near the desired outcome, the results have significantly improved. During the first iteration, the identification stage showed that replay needed 29 events to achieve the desired case ending. After initial adaptations, the replay execution needs only 21. This is already an indicator that the congruity is improving, as that means that there are less log events that do not have a mapping to a process model activity and vice-versa. Additionally, we can see that there are now two synchronous moves on the activities *Create PO* and *Record Goods Receipt*. The trace, existing of 8 events in the event log, now has a fitness value of 0.19 as opposed to 0 before adaptation.

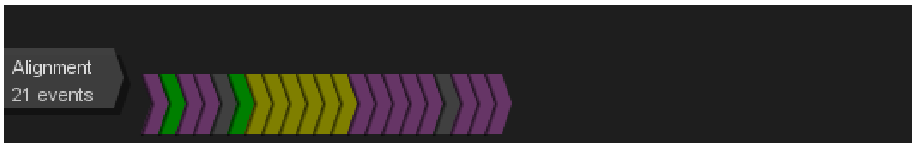
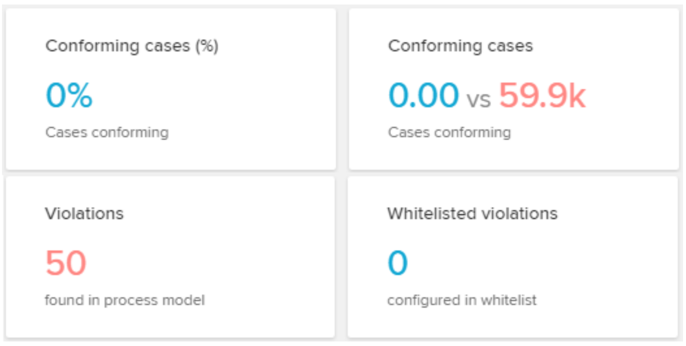


Figure 6.7: Initial results of adaptation, alignment on case 500450022286600030

As expected, the first iteration of the systematic approach does not yet yield the desired results. Therefore, several iterations have been performed in order to achieve results in which the conformance results generated prove to be of adequate quality. Table 6.1 describes what adaptations have been performed in every iteration, as well as the effect that these adaptations had on the fitness of the log-model pair.

Table 6.1: Adaptations and fitness for each iteration

Iteration	**Adaptations**	**Avg. fitness**	**Case fitness**
0	None	0	0
1	Log change: event name for PO creation and Goods Receipt Model change: inserted loop for Goods Receipt Log change: deleted several noise-inducing events (VIM — ..)	0.20	0.19
2	Model change: lowered abstraction level of SAP release strategy Log change: event name for Invoicing	0.50	0.47
3	Log change: changed timestamp for Goods Receipt to 23:59:59	0.59	0.59
4	Model change: removed end event	0.63	0.69
5	Model change: release PO only needed after approval Model change: allow VIM — Indexed to be repeated Log change: changed timestamp for Invoice Receipt to 23:59:59	0.90	0.90

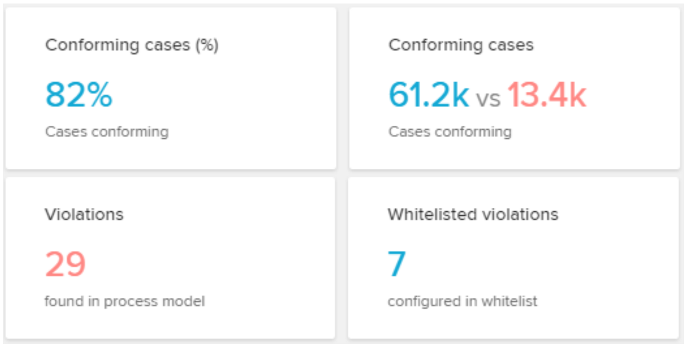

Conforming cases (%)
0%
Cases conforming

Conforming cases
0.00 vs 59.9k
Cases conforming

Violations
50
found in process model

Whitelisted violations
0
configured in whitelist

Figure 6.8: Initial conformance results in Celonis

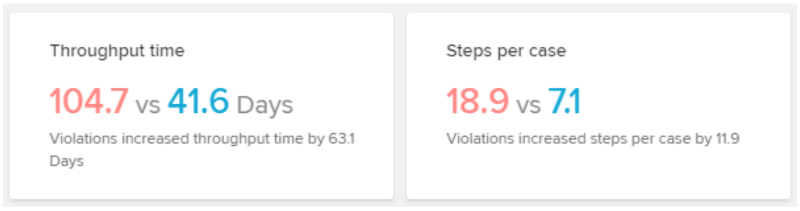

Conforming cases (%)
82%
Cases conforming

Conforming cases
61.2k vs 13.4k
Cases conforming

Violations
29
found in process model

Whitelisted violations
7
configured in whitelist

Figure 6.9: Conformance results in Celonis after applying systematic approach


Throughput time
104.7 vs 41.6 Days
Violations increased throughput time by 63.1 Days

Steps per case
18.9 vs 7.1
Violations increased steps per case by 11.9

Figure 6.10: Generated insights by Celonis

Chapter 7

Conclusion

The aim of this research was to develop a systematic approach that enables relevant stakeholders to perform a process conformance analysis on input that is considered not to be directly suitable. Incongruous input does not provide any reliable results when applying conformance checking techniques. Therefore, it is needed to make adaptations to the event log and process model in order to generate meaningful insights into the conformance of a business process execution. The research goal stated in Chapter 1 was the following:

> *To develop a systematic approach for performing conformance checking on incongruous input.*

The research problem was tackled by using the CRISP-DM cycle, starting with developing a business understanding of the problem at hand. This business understanding was developed by talking to experts in the field of internal audit (risk, control) at both Protiviti and Wessanen, as well as applying conformance checking to unsuitable data. Additionally, this provided an understanding of the data available. By creating an example dataset, it was possible to perform the next steps in the CRISP-DM cycle: data preparation and modelling. The evaluation of the developed systematic approach was done by applying the approach in a case study, using process data retrieved from the ERP system of Wessanen.

Results of the case study conclude that it is possible to apply conformance checking techniques (primarily log replay) on a log-model pair that is incongruous. Conformance checking can be performed when applying the added input of domain knowledge—process, data, and analysis expertise—in a systematic and iterative manner. The systematic approach developed as part of this research has shown to be effective in the case study of this work, and could be an excellent starting point for conformance checking analyses in practice. Applying the systematic approach in the case study has improved the average fitness from 0.00 to 0.90 on all cases, allowing analysts to draw conclusions about the conformity of process executions to their intended process flow.

7.2 Contribution

This work has primarily contributed to the usage of conformance checking techniques in practice. As discussed in Chapter 1, it is generally the case that the input for conformance checking is not ideal to generate good results. By using the systematic approach in this work, it is possible to do data-driven conformity analyses that generate meaningful insights that can be used to define the effectiveness of relevant process controls. This can greatly benefit many organizations by helping them improve their business processes. With this approach, all process executions (full population) can be analyzed, as opposed to using traditional sampling methods.

This work has also contributed to research on the broad topic of conformance checking. It has built on the work of Rogge-Solti et al. (2016), which had previously introduced the term 'generalized conformance checking'. The definition given in their work is performing conformance checking on a process model and event log of which neither is fully reliable. Building on this definition, this work has made an effort to make a step in the direction of generalized conformance checking. This is done by developing the systematic approach that allows stakeholders to adapt the event log and process model such that they become *congruous*.

7.3 Limitations and Future Work

This research has been conducted under some assumptions that have to be pointed out to clarify the contribution to the field of conformance checking. This section discusses those assumptions. Although the results have shown that the systematic approach can be used to generate relevant insights into process conformance, there also exist some limitations to its applicability. Future research should be conducted on this topic for it to become fully mature over the years.

Before getting into limitations, this work has been developed under the assumption that both the event log and the process model at hand are (partially) reliable. They are based on the same process and both hold *value* that is desired in the conformance analysis. In practice, it could be the case that the process model available is completely different from the ideal process model in the context of the conformance analysis. In such a scenario, the analyst is likely to be better off creating a new process model from scratch or discovering the model from the event log and altering it where needed, rather than identifying and overcoming incongruity issues with the developed approach.

A big limitation is that the systematic approach requires a large amount of manual labor in the identification of congruity issues, and the adaptation of the event log and process model. Primarily the identification of congruity issues could benefit from having algorithms that can automatically detect problems that occur in the log-model combination. It is likely that this process could be sped up by using machine learning techniques (such as pattern recognition and natural language processing) to make this process (partly) automatic. The analyst could be pointed at likely incongruities that are relatively easy to detect, such as naming misalignments and parts of the process that often do not adhere to the process model. The adaptation of the process model and event log could also be partially automated by directly applying the activity name changes and changing model structures but this is a more complex problem due to the domain knowledge needed to judge whether elements of the process can be left out or adapted in some way.

An assumption that is made in this work is that the conformance analysis has a focus on finished cases. This means that the full event log has been filtered on the cases that have reached a particular end state in the process. As the wish for continuous monitoring grows in modern organizations Minnaar, D., Littley, J. Farineau (2008), it is very relevant to also look at cases that have not yet reached their final destination in the process. This could be done by inserting a cut-off activity after the last activity in the trace of unfinished cases, that directly brings the process to the final state in the process model. This would ensure that the log replay technique sees the case as being unfinished, but allowing it to play without any nonconformities due to the case missing needed activities.

In the case study of this work, we came across the need for process flows that were *k-bound.* More specifically, we came across the intended process structure that made for traces such as $\langle a, b, b, c, c, d \rangle$ and $\langle a, b, b, b, c, c, c, c, d \rangle$, in which one activity has to occur exactly the same time as another (in this example $\#a = \#b$). Ramezani Taghiabadi (2017) discusses this problem in some more detail. Structures such as these are relatively common but are hard to check for conformance with current techniques. New conformance checking techniques should be developed to account for processes that require *boundedness* to the number of activities.

Another limitation is that this research has a focus on process executions following a particular order of activities, as described in the process model. The event data, such as the resource used or the time spent between activities, is not taken into account. In practice, the underlying event data is often just as important as the process flow itself. For example, segregation of duties is an important internal control measure that cannot be detected by using the methods covered in this work. Current conformance checking techniques are quite limited in their ability to create *rules* for process event data to adhere to. Progressing in this area is something that would greatly benefit the industry. Mannhardt et al. (2016) discusses this problem in more detail.

Finally, a limitation in this work and in the industry in general is that the process models available are not of a high quality, especially for the sake of conformance checking. Future work could focus on generally expanding on the topic of generalized conformance checking to solve this problem. The industry could greatly benefit from conformance checking techniques that are more robust, allowing for input that is not perfectly congruous with the event log at hand. Again, pattern recognition techniques in machine learning could be highly effective at uncovering process nonconformities, even as they occur in real-time.

Bibliography

Accorsi, R. (2011). An Approach to Data-driven Detective Internal Controls for Process-aware Information Systems. *Data Usage Management on the Web.* 1

Adriansyah, A., van Dongen, B. F., and van der Aalst, W. M. P. (2011). Towards Robust Conformance Checking. In *Lecture Notes in Business Information Processing.* 5

Buijs, J. C. A. M., van Dongen, B. F., and van der Aalst, W. M. P. (2012). On the role of fitness, precision, generalization and simplicity in process discovery. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7565 LNCS(PART 1):305–322. 5, 19

Buijs, J. C. A. M., van Dongen, B. F., and van der Aalst, W. M. P. (2014). Quality Dimensions in Process Discovery: The Importance of Fitness, Precision, Generalization and Simplicity. *International Journal of Cooperative Information Systems.* 27

Claes, J., Vanderfeesten, I., Reijers, H. A., Pinggera, J., Weidlich, M., Zugal, S., Fahland, D., Weber, B., Mendling, J., and Poels, G. (2012). Tying Process Model Quality to the Modeling Process: The Impact of Structuring, Movement, and Speed. In *Proceedings of the 10th International Conference on Business Process Management (BPM 2012)*, pages 33–48. 5, 26

Consultancy (2017). Protiviti Nederland, Retrieved October 10, 2018 from *https://www.consultancy.nl/adviesbureaus/protiviti*. 3

COSO (2013). Internal Control Integrated Framework. *Committee of Sponsoring Organizations of the Treadway Commission*, (May):10. 14

Dumas, M., La Rosa, M., Mendling, J., and Reijers, H. A. (2013). *Fundamentals of Business Process Management.* ix, 3, 4

Fahland, D. and van der Aalst, W. M. P. (2012). Repairing process models to reflect reality. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* 21

Fahland, D. and van der Aalst, W. M. P. (2015). Model repair - Aligning process models to reality. *Information Systems*, 47:220–243. 5

Flinders, M. and Denton, M. (2008). Internal Control. In *Delegated Governance and the British State: Walking without Order.* 11

Forbes (2015). What IT Needs to Know About the Data Mining Process, Retrieved July 8, 2018 from *https://www.forbes.com/sites/metabrown/2015/07/29/what-it-needs-to-know-about-the-data-mining-process/*. 6

IEEE Task Force on Process Mining (2011). Process Mining Manifesto. *Business Process Management Workshops*, pages 169–194. 16, 25, 65

Inside Big Data (2017). (2017). The Exponential Growth of Data, Retrieved September 20, 2018 from *https://insidebigdata.com/2017/02/16/the-exponential-growth-of-data/*. 1

Kalenkova, A., Burattin, A., de Leoni, M., van der Aalst, W. M. P., and Sperduti, A. (2016). Discovering High-level BPMN Process Models from Event Data. *Tech. rep. BPM Center Report BPM-16-09*, page 46. 5

Kirchmer, M. (2011). High Performance Through Process Excellence. *Business.* 1

Kirchmer, M. (2016). The Process of Process Management: Enabling High Performance in a Digital World. In *Lecture Notes in Business Information Processing.* 1

Lu, X. (2013). Artifact-Centric Log Extraction and Process Discovery. (September):1–116. 45

Mannhardt, F., de Leoni, M., Reijers, H. A., and van der Aalst, W. M. P. (2016). Balanced multi-perspective checking of process conformance. *Computing.* 55

Minnaar, D., Littley, J. Farineau, D. (2008). Continuous Auditing and Continuous Monitoring: Transforming Internal Audit and Management Monitoring to Create Value. *Technical Report.* 54

Piessens, D. (2011). *Event Log Extraction from SAP ECC 6.0.* PhD thesis. 45

Protiviti. About Us, Retrieved July 7, 2018 from *https://www.protiviti.com/US-en/about-us*. 3

Ramezani Taghiabadi, E. (2017). *Understanding Non-Compliance.* PhD thesis, Eindhoven University of Technology, Eindhoven. ix, 1, 12, 13, 54

Ratcliffe, T. A. and Landes, C. E. (2009). Understanding Internal Control and Internal Control Services. 14

Rogge-Solti, A., Senderovich, A., Weidlich, M., Mendling, J., and Gal, A. (2016). In Log and Model We Trust? A Generalized Conformance Checking Framework. 9850:179–196. 2, 21, 54

Shearer, C. (2000). The CRISP-DM model: The New Blueprint for Data Mining. *Journal of Data Warehousing14.* 6

van der Aalst, W. M. P. (2011). Process Mining: Discovery, Conformance and Enhancement of Business Processes. 1, 5, 16, 18, 23

van der Aalst, W. M. P., Adriansyah, A., and van Dongen, B. (2012). Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery.* 20

van der Aalst, W. M. P. and Günther, C. (2007). Finding Structure in Unstructured Processes: The Case for Process Mining. *Proceedings - 7th International Conference on Application of Concurrency to System Design, ACSD 2007*, pages 3–12. 1

van der Aalst, W. M. P., van Hee, K. M., van der Werf, J. M., and Verdonk, M. (2010). Process Mining to Support Tomorrow ' s Auditor. *IEEE Computer Society*, 43(3):90–93. 1

van Dongen, B. F., Carmona, J., and Chatain, T. (2016). Alignment-based metrics in conformance checking. In *CEUR Workshop Proceedings.* 19

Weske, M. (2012). *Business Process Management: Concepts, Languages, Architectures.* 1

Wessanen (2018). Our History, Retrieved October 10, 2018 from *https://www.wessanen.com/about-us/our-history/*. 3

Appendix A

Running Example

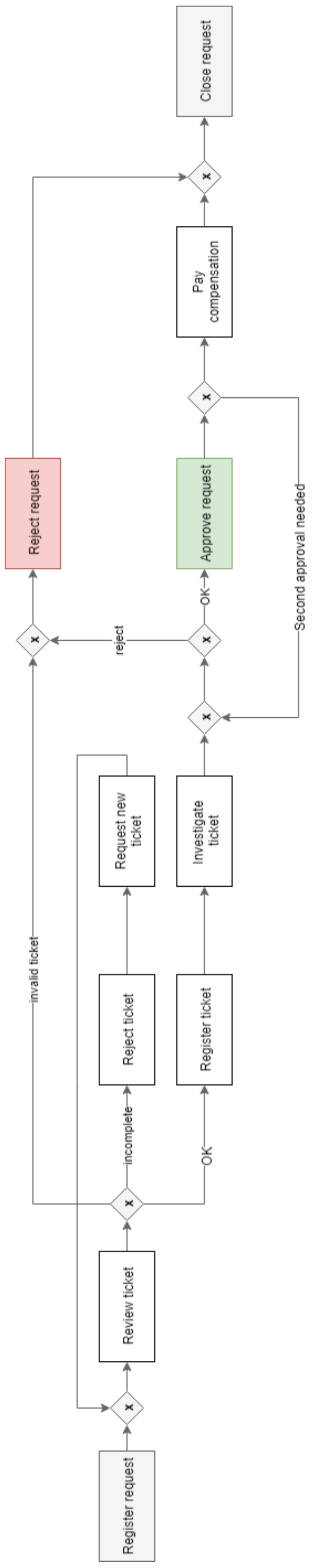


Figure A.1: Starting point for the running example process model

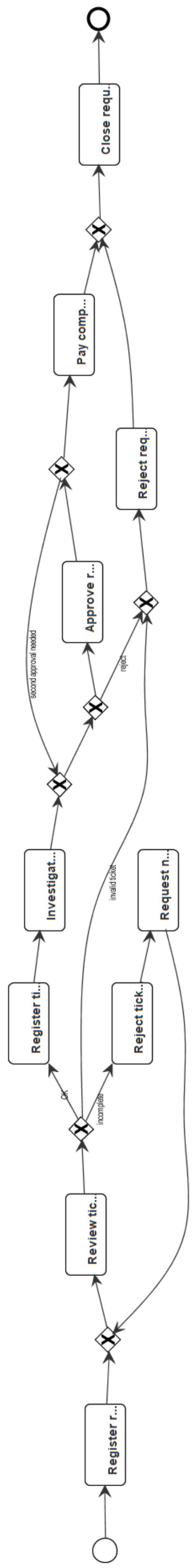


Figure A.2: Running example BPMN model

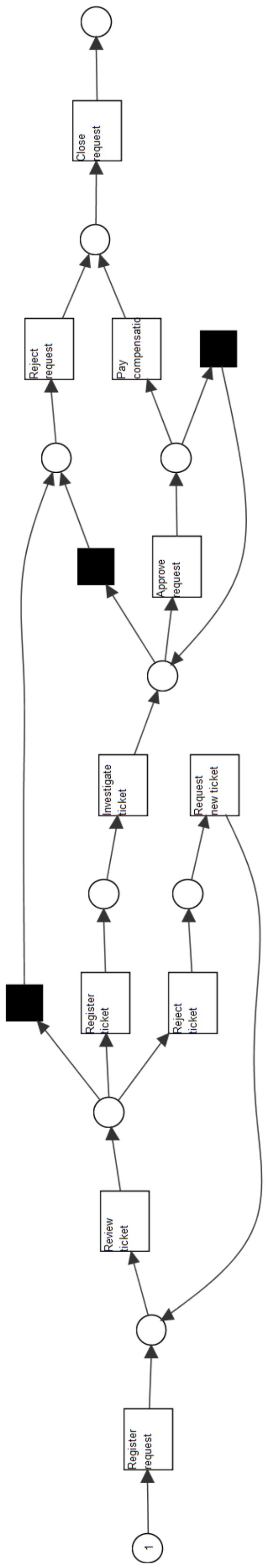


Figure A.3: Running example reduced Petri net

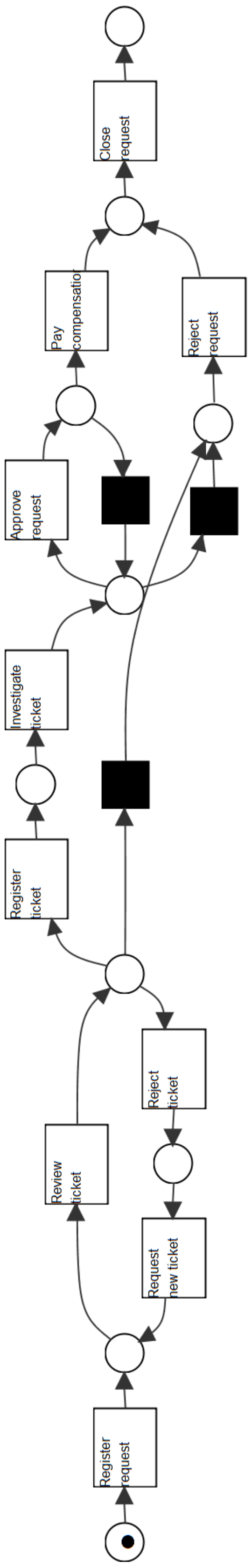


Figure A.4: Running example reduced Petri net with condensed formatting

Appendix B

Event Log Maturity Model

Table B.1: Maturity levels for event logs (from Process Mining Manifesto IEEE Task Force on Process Mining (2011))

Level	Characterization	Examples
⋆⋆⋆⋆⋆	Highest level: the event log is of excellent quality (i.e., trustworthy and complete) and events are well-defined. Events are recorded in an automatic, systematic, reliable, and safe manner. Privacy and security considerations are addressed adequately. Moreover, the events recorded (and all of their attributes) have clear semantics. This implies the existence of one or more ontologies. Events and their attributes point to this ontology.	Semantically annotated logs of BPMN systems.
⋆⋆⋆⋆	Events are recorded automatically and in a systematic and reliable manner, i.e., logs are trustworthy and complete. Unlike the systems operating at level ⋆ ⋆ ⋆, notions such as process instance (case) and activity are supported in an explicit manner.	Event logs of traditional BPM/-workflow systems.
⋆⋆⋆	Events are recorded automatically, but no systematic approach is followed to record events. However, unlike logs at level ⋆⋆, there is some level of guarantee that the events recorded match reality (i.e., the event log is trustworthy but not necessarily complete). Consider, for example, the events recorded by an ERP system. Although events need to be extracted from a variety of tables, the information can be assumed to be correct (e.g., it is safe to assume that a payment recorded by the ERP actually exists and vice versa).	Tables in ERP systems, event logs of CRM systems, transaction logs of messaging systems, event logs of high-tech systems, etc.
⋆⋆	Events are recorded automatically, i.e., as a by-product of some information system. Coverage varies, i.e., no systematic approach is followed to decide which events are recorded. Moreover, it is possible to bypass the information system. Hence, events may be missing or not recorded properly.	Event logs of document and product management systems, error logs of embedded systems, worksheets of service engineers, etc.
⋆	Lowest level: event logs are of poor quality. Recorded events may not correspond to reality and events may be missing. Event logs for which events are recorded by hand typically have such characteristics.	Trails left in paper documents routed through the organization (”yellow notes”), paper-based medical records, etc.

Appendix C

Case study

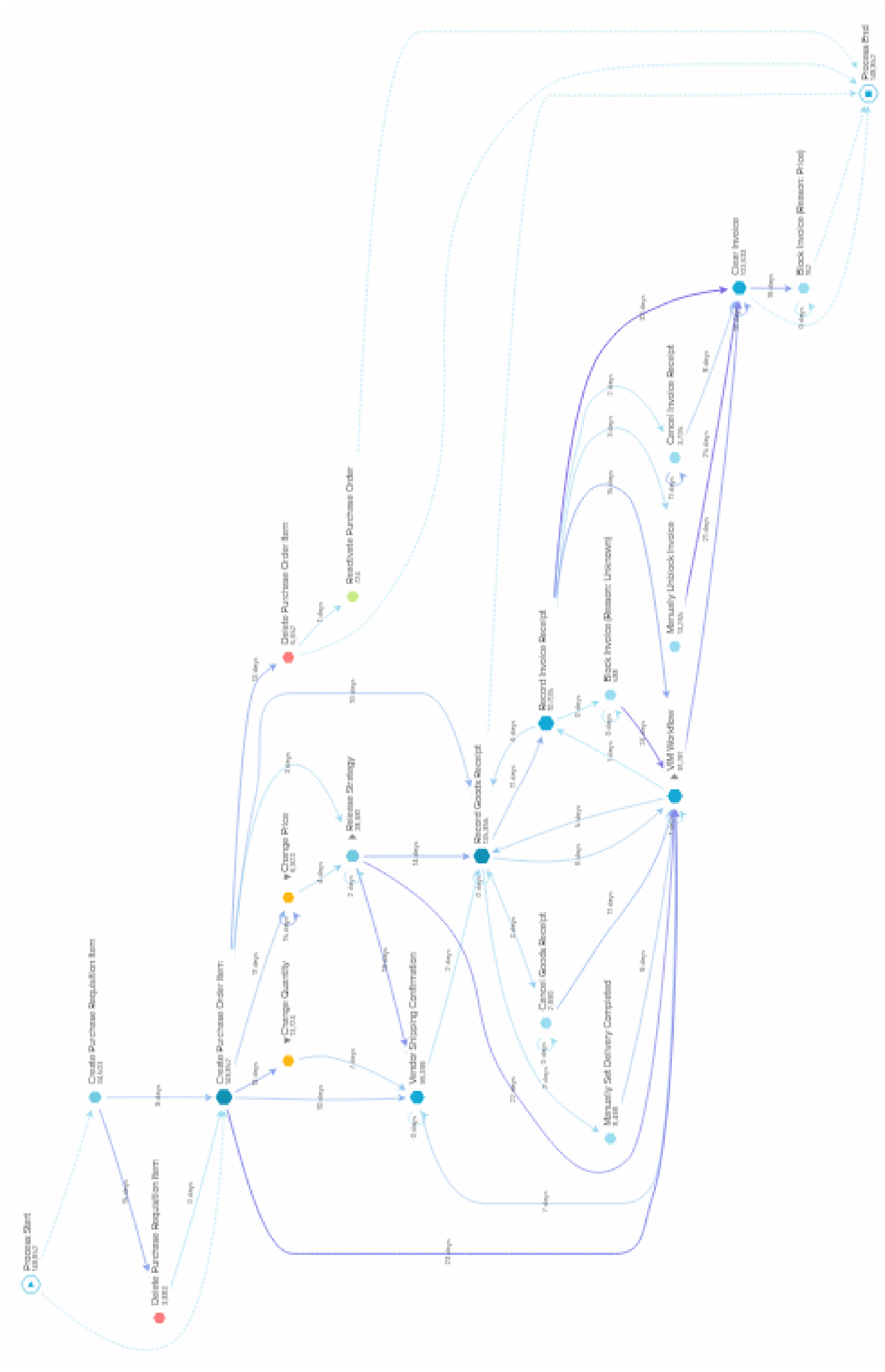

Process Start
Create Purchase Requisition Item
Delete Purchase Requisition Item
Create Purchase Order Item
Change Quantity
Change Price
Delete Purchase Order Item
Reactivate Purchase Order
Release Strategy
Vendor Shipping Confirmation
Record Goods Receipt
Cancel Goods Receipt
Manually Set Delivery Completed
Record Invoice Receipt
Block Invoice (Reason: Unknown)
VIM Workflow
Manually Unblock Invoice
Cancel Invoice Receipt
Clear Invoice
Block Invoice (Reason: Price)
Process End

Figure C.1: Process model discovered from Wessanen ERP data

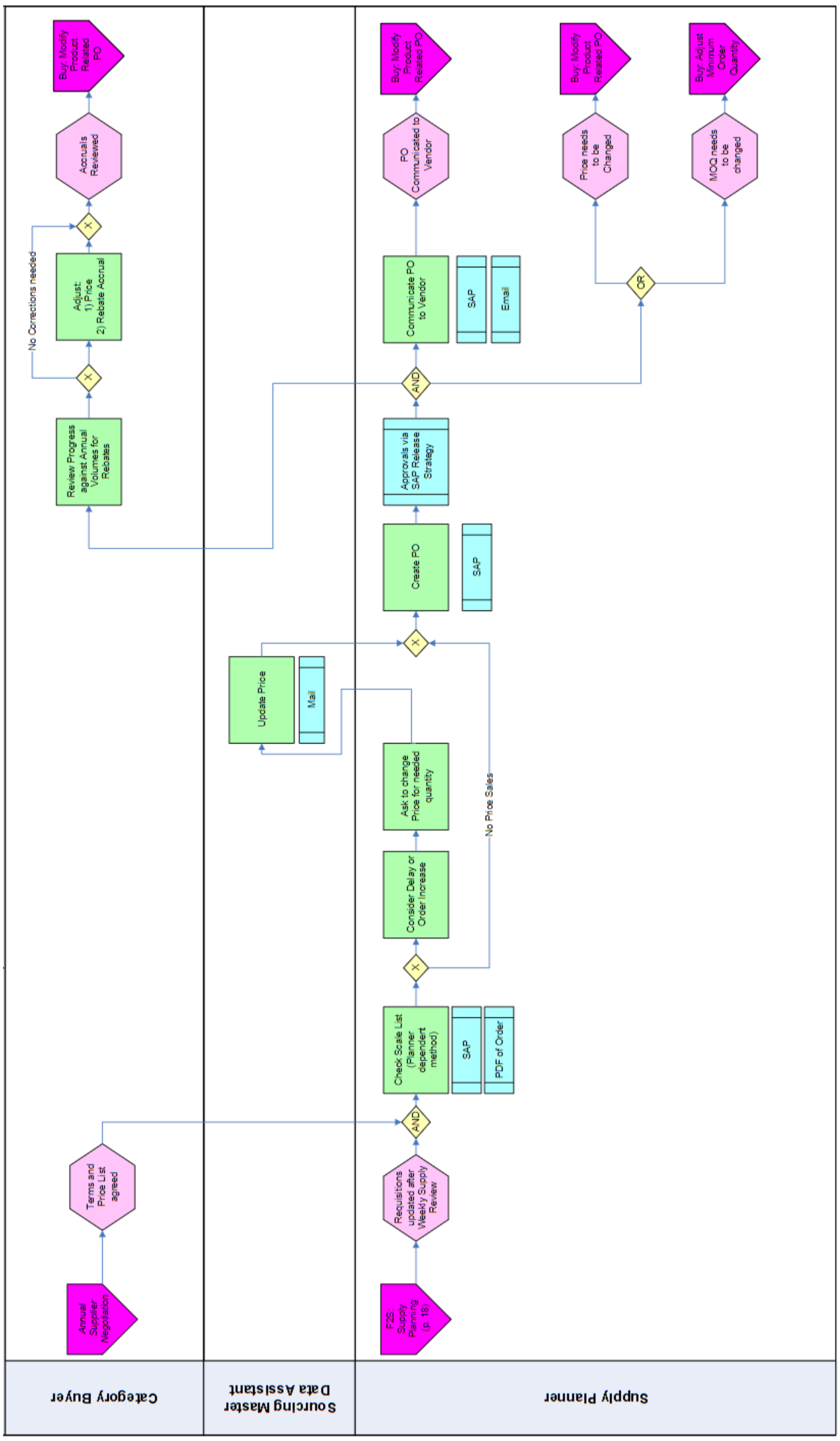

Category Buyer
Sourcing Master Data Assistant
Supply Planner
Annual Supplier Negotiation
Terms and Price List agreed
Review Progress against Annual Volumes for Rebates
X
No Corrections needed
Adjust: 1) Price 2) Rebate Accrual
X
Accruals Reviewed
Buy: Modify Product Related PO
Update Price
Mail
F2S: Supply Planning (p. 18)
Requisitions updated after Weekly Supply Review
AND
Check Scale List (Planner dependent method)
SAP
PDF of Order
X
Consider Delay or Order Increase
Ask to change Price for needed quantity
No Price Sales
X
Create PO
SAP
Approvals via SAP Release Strategy
AND
Communicate PO to Vendor
SAP
Email
PO Communicated to Vendor
Buy: Modify Product Related PO
OR
Price needs to be Changed
Buy: Modify Product Related PO
MOQ needs to be changed
Buy: Adjust Minimum Order Quantity

Figure C.2: Unadapted process model as provided by Wessanen

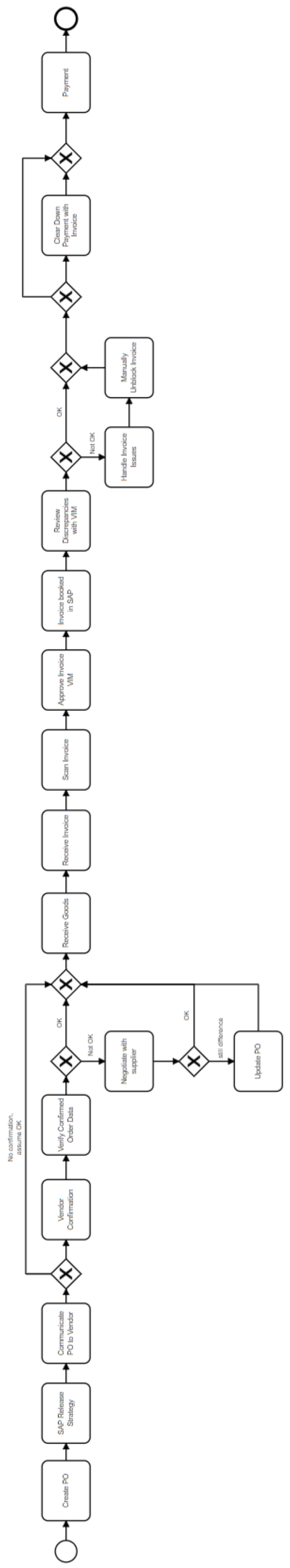


Figure C.3: Translated, unadapted process model

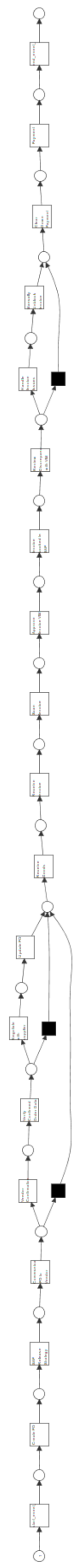


Figure C.4: Translated, unadapted process model as Petri net