



universität
wien

DISSERTATION / DOCTORAL THESIS

Titel der Disseratation / Title of the Doctoral Thesis

„TIDATE - Time and Data Aware Process Mining at Runtime“

verfasst von / submitted by

Florian Stertz

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Doktor der technischen Wissenschaften (Dr.techn.)

Wien, 2022 / Vienna, 2022

Studienkennzahl lt. Studienblatt /
degree programme code as it appears on
the student record sheet:

UA A 786 880

Studienrichtung lt. Studienblatt /
degree programme as it appears on
the student record sheet:

Informatik

Betreut von / Supervisor:

Univ.-Prof. Dipl.-Math. oec. Dr. Stefanie Rinderle-Ma

Declaration of Authorship

I, Florian Stertz, declare that I have authored this thesis entitled, “TIDATE - Time and Data Aware Process Mining at Runtime” independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Ich, Florian Stertz, erkläre an Eides statt, dass ich die vorliegende Arbeit mit dem Titel “TIDATE - Time and Data Aware Process Mining at Runtime” selbstständig verfasst, andere als die angegebenen Quellen / Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Signature/Unterschrift

Date

Acknowledgements

I would like to thank my supervisor Prof. Dr. Stefanie Rinderle-Ma for academical guidance during this whole thesis, even 355.77 kilometers apart. Through our regularly discussions, the ideas of this thesis could be formed and molded into scientific contributions. I also want to highlight, her swift and precise feedback, even in the darkest time of night, enabled me to never miss a deadline. I could not ask for better supervisor.

Furthermore, I thank Prof. Dr. Andrea Burattin, who has given me good ideas for papers and future work during discussions at conferences, and Prof. Dr. Erich Schikuta who accepted to act as referees.

While writing this thesis, I had the fantastic experience of working with great colleagues and friends at the WST team, who provided me with a good amount of laughs, academical insights, and, at least sometimes, productive working sessions.

Special thanks go to Karolin Winter for the great collaboration and discussions on our publication.

Moreover, I thank Jürgen Mangler, whose advice helped me, even in the unholy hours of the day and night and I thank Monika Hofer-Mozelt. Without her help, I would be stuck with some paperwork or still even on sick leave.

In addition, I want to thank my dearest friends, that play games with me for at least 15 years now. Together we created the funniest and most enjoyable hours of procrastination, I can imagine. Special thanks go to my best man, Philip Thonke, who attended my wedding, even though he had to take three flights to arrive here. Without all of you, my dissertation would have been finished at least 2 years ago, but the enjoyable moments we shared, make up for this tenfold.

Last, but not least, I want to thank my wonderful family. I cannot even imagine a life without them. I thank my children Leia, Lara and Finn for their love, every night they allowed me some sleep, diapers, and the joy they brought to my life. My dearest thanks go to my hot waifu Lisa, who has always been there for me, encouraged me, gave me loads of chores, and without her support, this would never have been possible.

Lastly, I want to thank, the new HR4U department. The new system is miserable and exhausting, which accelerated my desire to finish to this.

May the force be with us. Always.

Abstract

In a short period of time, process mining emerged successfully as a technology to support companies in identifying business processes, check the conformance of business processes, and even enhance business processes by detecting bottlenecks with shared resources across different processes. This is usually achieved by using process execution logs, i.e., data files containing information of the executed tasks like the name of a task, or the timestamp of the execution. In its traditional application, process mining is applied offline, i.e., after the execution. It is also important to note, that specific data elements, e.g., time series sensor data, that are often captured outside of the process execution log, can potentially impact the execution of a process as well, but are not properly taken into account in traditional process mining, e.g., the blood pressure and other vital signs are periodically measured for patients in a hospital, but the data points are not related to a specific task in a process. Online process mining tackles the first problem of traditional process mining and applies techniques directly during the execution, but data elements are often not taken into account as well as exterior data, like sensor data elements. These limitations of offline process mining and negligence of exterior data, create a research gap, i.e., how can deviations in the execution of process instances be explained? E.g., a new medical guideline changes the therapy plan for patients in a hospital. Online process mining using data elements can detect a new process model in store it in a history of older process models for a process. This allows domain experts to determine the type of concept drift and suggest adjustments for the process model for future instances. When process mining techniques take exterior data into account, an explanation for a deviation can be provided for domain experts, i.e., a specific drug is not working according to the body temperature of a patient, hence a different drug is used. The body temperature is usually monitored constantly and not logged in a process execution log directly, but offers an explanation for the change in the therapy plan. This thesis provides the TIDATE framework, containing novel concepts and algorithms for online process mining. It comprises techniques to generate an event stream, i.e., a data structure created during the execution of business processes containing all generated events of process instances. In addition, since business processes change constantly, TIDATE also provides an innovative concept, that reflects the evolution of a business process, containing all different process models, discovered after drifts in the workflow perspective and/or data perspective for a process, a **process history**. The online conformance checking approach of this framework is based on data elements, interior data captured by the process execution as well as exterior data like sensor data streams, to identify the source of a potential drift in the business process and weigh deviations in a process instance to better quantify the degree of conformance of an instance. These concepts are evaluated through prototypical implementations using artificial and real world data sets. To conclude TIDATE enables companies to generate event streams and apply online process mining techniques. It is used to identify drifts in the process logic as early as possible, find the reason and explanation of the drift and to detect and quantify non conformance in the behavior of executed process instances.

Kurzfassung

In einer relativen kurzen Zeit hat sich **Process Mining** erfolgreich als Technologie etabliert, welche Unternehmen unterstützt **Business Processes** zu finden. Process Mining Algorithmen kontrollieren auch die Ausführung von Prozessinstanzen im Vergleich mit einem Prozessmodell und das Verbessern der Prozesse durch das Entdecken von Engpässen bei der Nutzung von geteilten Ressourcen. Die Grundlage für diese Verfahren bilden **Process Execution Logs** Dateien. Diese beinhalten Informationen von bereits ausgeführten Aktivitäten alter Prozessinstanzen, wie Namen und Ausführungsdatum einer Aktivität. Üblicherweise werden Prozessminingalgorithmen nach der Exekution von Prozessinstanzen durchgeführt, **offline**. Außerdem werden Datenelemente, wie Sensordaten, welche nicht direkt in der Logik von Prozessen verarbeitet und gespeichert werden, nicht in die Algorithmen miteinbezogen, obwohl diese den Ablauf von Prozessinstanzen beeinflussen können. So werden z.B. der Blutdruck oder andere Vitalitätszeichen in bestimmten Abständen im Krankenhaus gemessen werden, die Messwerte werden jedoch nicht einer bestimmten Aktivität zugeordnet in einem Prozess. **Online** Prozessminingalgorithmen hingegen werden direkt während der Ausführung von Prozessinstanzen angewendet. Externe Datenelemente werden aber weiterhin nicht beachtet. Diese Limitierungen von offline Prozessminingalgorithmen, sowie die Vernachlässigung von externen Daten bilden eine Forschungslücke, z.B. wie kann man Abweichungen nicht nur feststellen sondern auch den Grund erklären? Sollte zum Beispiel eine neue medizinische Richtlinie den Therapieplan von Patienten in einem Krankenhaus ändern, so können online Prozessminingalgorithmen, welche Datenelemente miteinbeziehen, eine Veränderung in dem Prozessmodell feststellen. Durch das Entdecken aller Prozessmodelle für einen Prozess in seiner Lebenszeit, können die Arten von **Concept Drifts** (Änderungen im Modell) genau bestimmt werden und Domainexperten können diese Modelle nutzen um zukünftige Prozessinstanzen anzupassen. Durch in Betracht ziehen von externen Datenelementen kann der Grund für eine Abweichung festgestellt werden. Wird z.B. ein anderes Medikament verabreicht weil ein Medikament zuvor nicht gewirkt hat, kann dies über das Einbeziehen von externen Datenelementen, wie der Temperatur eines Patienten, ermittelt werden. Die Temperatur wird ständig gemessen, jedoch nicht als Datenelement in einer Aktivität gespeichert, aber sie würde den Grund für die Abweichung in der Prozessinstanz erklären im Therapieplan. Diese Abschlussarbeit präsentiert das TIDATE framework, welches neue Konzepte und Algorithmen für online Prozessmining beinhaltet. Es beinhaltet Techniken um einen **Event Stream** zu erzeugen, eine Datenstruktur welche direkt während der Ausführung einer Prozessinstanz generiert wird. Zusätzlich, da die Logik von Prozessen sich laufend ändert, bietet TIDATE auch ein innovatives Konzept, **Process History**, welches die Evolution von einem Prozess abbildet und Prozessmodelle beinhaltet, welche nicht nur die Arbeitsabläufe darstellen, sondern auch die Datenelemente berücksichtigen. Die online **Conformance Checking**-Algorithmen in TIDATE basieren auf den Datenelementen, welche intern gespeichert in einer Prozessengine sind und jene welche außerhalb der Prozesse gespeichert werden, wie z.B. Sensordaten. Das ist wichtig, um den Grund für Abweichungen in Prozessinstanzen zu ermitteln, aber

Kurzfassung

auch um die Abweichungen zu gewichten. Diese Konzepte sind evaluiert durch Prototypen, welche künstlich erzeugte Datensätze und auch echte Datensätze verwenden. Abschließend ist festzuhalten, dass TIDATE es Unternehmen ermöglicht, einen Datensatz zu generieren auf welchen online Prozessminingalgorithmen angewendet werden können. Auf diesen können Abweichungen und Änderungen in der Prozesslogik ehestmöglich erkannt, bestimmt und gewichtet werden.

Contents

Declaration of Authorship	i
Acknowledgements	iii
Abstract	v
Kurzfassung	vii
List of Tables	xiii
List of Figures	xv
List of Algorithms	xix
Listings	xxi
1 Introduction	1
1.1 Motivation	1
1.2 Problem Statement and Challenges	4
1.3 Research Questions	8
1.4 Methodological Background and Contributions	10
1.5 Thesis Structure	15
2 Related Work	17
2.1 Input Sources for Process Mining	18
2.2 Process Discovery Algorithms	19
2.3 Conformance Checking Algorithms	23
2.4 Process Enhancement	25
2.5 Concept Drift in Process Mining	26
3 Creating Data Sets for Process Mining	29
3.1 Data Set properties and format	32
3.1.1 Designing the XES-YAML Approach	33
3.1.2 Evaluation of XES-YAML File Format	37
3.1.3 Results of the XES-YAML Approach	38
3.2 Creating Process Mining Data Sets during the Execution of a Process . .	39
3.2.1 Process model creation	39
3.2.2 Implementation	40
3.2.3 Application Scenarios	42
3.3 Creating Accurate Data Sets for Process Mining with Human Resources Involved	42
3.3.1 Care scenario and contributions	43

3.3.2	Background on NFC technology	45
3.3.3	Conceptual solution design based on use cases	46
3.3.4	Realizing automatic task completion in a process-aware care solution	52
3.3.5	Practical evaluation	57
3.3.6	Discussion	62
3.4	Conclusion and outlook	64
4	Discovering the Evolution of Processes through Concept Drifts	67
4.1	Concept Drifts at the Control Flow Perspective	69
4.1.1	Fundamentals of Process Histories	70
4.1.2	Algorithms for Synthesizing Process Histories	72
4.1.3	Evaluation of process histories	78
4.1.4	Summary and Outlook of Process Histories	81
4.2	Concept Drifts on the Data Perspective	82
4.2.1	Fundamentals for Detecting Data Drifts	83
4.2.2	Detecting and Identifying Data Drifts	84
4.2.3	Evaluation of Data Drift Detection Algorithms	88
4.2.4	Conclusion for Data Drift Detection	89
4.3	Conclusion and Outlook	89
5	Time & Data-Aware Conformance Checking and Explaining Drifts	93
5.1	Extending Conformance Checking Algorithms Using an Advanced Cost Function	96
5.1.1	Advanced Cost Function	98
5.1.2	Evaluation of Extended Conformance Checking	103
5.2	Temporal Perspective	110
5.2.1	Temporal Conformance Checking	111
5.2.2	Evaluation of Temporal Conformance Checking	115
5.2.3	Financial Example	116
5.2.4	Manufacturing Example	118
5.3	Discover and Explain Concept Drifts based on external data sources . . .	121
5.3.1	Fundamentals of Dynamic Time Warping	122
5.3.2	Time Sequence Assignment and Root Cause Detection	124
5.3.3	Evaluation of Drift Explanation Discovery using Real-World Data .	128
5.3.4	Discussion of Drift Explanation Discovery	134
5.4	Conclusion and Outlook	135
6	Evaluating TIDATE - Time and Data Aware Process Mining at Runtime	137
6.1	TIDATE - Artifact Design	138
6.1.1	Methodology	139
6.1.2	Research Design	141
6.1.3	Artifact Validation	142
6.1.4	Research Execution	142
6.1.5	Discussion of Results	145
6.2	Expectations and Experiences of Process Mining in Action	147
6.2.1	Overview on Methodology and Study Design	149
6.2.2	Applied Process Mining Scenario	150

6.2.3	Results of Focus Group Interviews	152
6.2.4	Discussion and Implications for Research and Practice	155
6.3	Conclusion and Outlook	158
7	Conclusion	161
7.1	Results of Research Questions	161
7.1.1	Creating Data Sets for Process Mining	161
7.1.2	Discovering the Evolution of Processes through Concept Drifts . .	163
7.1.3	Time & Data-Aware Conformance Checking and Explaining Drifts	164
7.1.4	Evaluating TIDATE - Time and Data Aware Process Mining at Runtime	165
7.2	Future Work	165
	Bibliography	167

List of Tables

1.1	Example of data elements on the instance level for Table 1.2	4
1.2	Example of process execution log for the example from Figure 1.1. Each id is matching an id from Table 1.1.	5
1.3	Details of Chapter 3	12
1.4	Details of Chapter 4	13
1.5	Details of Chapter 5	14
2.1	Dependency relation between Tasks for the dependency graph in Figure 2.3	22
2.2	Example of alignment with only synchronous moves.	24
2.3	Example of alignment with a model move.	24
2.4	Example of alignment with a log move.	24
3.1	Comparison of different log formats for XES serialization. Well structured reflects if the data format can easily be checked for a specific format. Comments reflects the ability to make comments in the log file. Append without Parsing describes if the format is able to be extended without parsing the complete file format.	34
3.2	Performance Cornerstones	38
3.3	Use case: Register NFC tag	47
3.4	Use case: Register resident	48
3.5	Use case: Write on NFC tag	49
3.6	Use case: Write utility information	49
3.7	Use case: Get ToDo list	50
3.8	Use case: Give utility to resident	50
3.9	Use case: Document further information	51
3.10	Use case: Document injected task	51
3.11	Use case: Delete resident	52
3.12	Possible improvements through automatic documentation	59
5.1	Alignment of the first deviation	108
5.2	Alignment of the second deviation	108
5.3	Results Alg. 5	109
5.4	BPIC 2012: Task Duration in seconds for all 6 events of 3271 process instances	117
5.5	BPIC 2012: Temporal Distance in Seconds of the first 10469 process instances with κ set to 200	118
5.6	Manufacturing: Task Duration in Seconds of first 30 process instances in the data set.	121
5.7	Results of both Algorithms	132
5.8	Runtime of Alg.8.	134
5.9	Runtime of Alg.9.	134

List of Tables

6.1	Focus Group Participants Profile	153
-----	--	-----

List of Figures

1.1	Example of a process model describing the treatment for gall stones designed after description from https://www.gesundheit.gv.at/krankheiten/verdauung/gallenblase/gallensteine-therapie	3
1.2	Overview of Targets, Research Goals, Problems and where addressed in this thesis.	8
2.1	Process Model from Figure 1.1 with abbreviated labels for an easier understanding.	19
2.2	Example of process execution log and event stream for the process model in Figure 1.1	20
2.3	Example of dependency graph depicting the frequency of sequences and the dependency relation from Table 2.1	22
2.4	Example of the splits for the model in Figure 2.1. Following a similar design to depict the splits as seen in [LFvdA13]	23
2.5	Example of a small social network for the bottom path of the process model depicted in Figure 2.1.	26
2.6	Example of an incremental drift to the process model in Figure 2.1. The red dashed line marks the additional event.	27
2.7	Example of a recurring drift. Two process models alternate depending on the season.	27
3.1	Overview of Targets, Research Goals, Problems of this thesis. The features for this chapter are marked.	29
3.2	Example Process containing 2 parallel branches with 3 activities each in a loop.	37
3.3	Time consumption for storing events in XES-XML and XES-YAML	38
3.4	Process model containing one decision and one parallel gateway (modeled in BPMN, using Signavio)	39
3.5	CPEE Process model	40
3.6	Daily morning care routine (BPMN notation using Signavio)	43
3.7	Resident bed equipped with NFC reader and a NFC tag on a small plastic lid.	44
3.8	Use case Diagram showing the potential actors and use cases	47
3.9	Use case analysis: derivation of components and data artefacts	53
3.10	Architecture for process-based care system with embedded NFC-based documentation. Red-dotted components have been developed in this work.	54
3.11	The communication of the components.	57
3.12	Process Model for Afternoon / Evening Routine	58
3.13	Paper-based documentation	61
3.14	XES log (log-based documentation)	62

4.1	Overview of Targets, Research Goals, Problems of this thesis. The features for this chapter are marked.	67
4.2	Small process model example based on the running example in Figure 1.1. The activities are executed in sequence.	70
4.3	Small process model example based on the running example in Figure 1.1. The activities are executed in parallel.	70
4.4	Event stream containing two traces with different order of events.	71
4.5	A process model with one parallel gateway and 3 related traces. While the Move-Log fitness is perfect for the first two traces, the last trace contains an additional event and receives a lower score	73
4.6	Model M_{n-1} is only fitting the first trace perfectly. Model M_n is fitting all traces. M_n is now the new model.	75
4.7	Complete process history of a single business process	77
4.8	Process Models containing concept drifts.	80
4.9	The concept drift from a to b is not detectable as well as the drift from a to c, since traces from b and c fit a. The drift from b to c is detectable.	80
4.10	Process Models containing concept drifts.	81
4.11	Process model with data attributes of event Transportation	82
4.12	Process History showing a data drift in the attribute speed	84
4.13	Synthesising a process history with $\kappa = 1$ and $\phi = 1$	86
4.14	Results reflecting the range of the torque value	88
5.1	Overview of Targets, Research Goals, Problems of this thesis. The features for this chapter are marked.	93
5.2	An overview of the three different areas of this chapter. Conformance checking on a structural perspective is targeting the order of event. Semantic conformance checking is focusing on other perspectives, i.e., the temporal perspective. The last area, aims at taken external data into account to discover the source of a drift in a process model.	94
5.3	A smaller version of the running example depicted in Figure 1.1. Here only the tasks for patients with severe symptoms are used for demonstration.	95
5.4	Conformance Checking Types and Affected Artifacts. This section focuses on the circled area.	96
5.5	Scenarios for Conformance Deviations (PM: Process Model)	97
5.6	Example for Numerical Values. A range classifying acceptable values	101
5.7	Example for Categorical Values	102
5.8	Example for Time Sequence Values. The average time sequence and the maximal allowed distance to it are stored	103
5.9	Artificial Example	105
5.10	Manufacturing Example. The 'S' in the modeled tasks, shows that a script is executed by the process execution engine after a task has been completed.	106
5.11	Shop Floor: Robot, Lathe, Micrometer, Stock	107
5.12	GV12 Part [SRM20a]	107
5.13	Trays in Stock Area [SRM20a]	107
5.14	Time Sequences of Manufacturing Example. Red line is the average time sequence, dashed from one trace. The cost is reduced because the distance is below the threshold of 29.12	108

5.15	Conformance Checking Types and Affected Artifacts. This section focuses on the circled area.	110
5.16	Temporal Profile – Example.	112
5.17	Process Model Infused with Temporal Profile – Example	116
5.18	Manufacturing of Parts	119
5.19	Conformance Checking Types and Affected Artifacts. This section focuses on the circled area.	121
5.20	Concept drift resulting in adapted process model – medical example . . .	123
5.21	Plot of two time sequences and their corresponding values	123
5.22	Warp matrix constructed using DTW: orange cell = distance	124
5.23	Proposed architecture: red parts denote the contribution of this section . .	125
5.24	Exemplary Result of Alg. 8 The red line is the average sequence calculated using DBA. The green dashed lines represent outliers, potentially due to a faulty process instance.	127
5.25	GV12 part	129
5.26	Batches of GV12 parts	130
5.27	Result of implementation. The red line represents the average sequence . .	131
5.28	Chips on GV12 - wrong measurement	132
5.29	GV12 Prototype Part	133
6.1	Overview of Targets, Research Goals, Problems of this thesis. The features for this chapter are marked.	137
6.2	TAR cycles (based on [Wie14])	139
6.3	TIDATE architecture	141
6.4	Process model used in the process execution engine for producing Turm parts.	143
6.5	Small example of process model in execution engine at client. The time $\sim T$ represents the expected task duration on average.	144
6.6	Process model for the daily production of parts. The signal indicates a new online mining item.	147
6.7	Double layer focus group study. All participants are grouped along both layers.	149
6.8	Electroplating – A bath for surface treatment of parts has to be refilled after use or time	151

List of Algorithms

1	Algorithm for synthesizing a process history based on an event stream. . .	74
2	Algorithm for identifying the specific type of concept drift.	79
3	Algorithm to synthesise a process history	91
4	Algorithm to identify data drift.	92
5	Finding Cost of Alignment	100
6	Temporal Profile Generation	113
7	Finding Cost of Alignment	115
8	Find relevant time sequences and compute avg. time sequence	126
9	Detecting a set of sensor data streams which caused a drift	127

Listings

3.1	Example XES log file snippet for one process instance with severe symptoms for process depicted in Figure 1.1	30
3.2	Example XES Event Stream for one process instance with severe symptoms for process depicted in Figure 1.1	31
3.3	Example of a YAML XES File	35
3.4	Pseudocode for loading a XES-YAML file Into Memory	36
3.5	Example of storing received data	41
3.6	Example of Log File	41

1 Introduction

“*Business processes represent a core asset of corporations*” [DRMR18]. They describe the connection and interaction of resources of an organization within and with external resources from other organizations. The area of analyzing business process and checking their behavior is defined as **Business Process Management** (BPM). Driven by the ever increasing number of recorded events in working processes, i.e., the tasks performed in a treatment plan for a patient in a nursing home or tasks performed by employees and robots in a manufacturing process, process mining enables companies to gather and visualize information from the actual executed processes instead of information from the design of a process [vdA16]. Process mining enables companies to analyze their business processes by discovering process models (referred to as **process discovery**) [LFvdA13, VdAWM04, WvDADM06], to monitor the conformance of business processes (referred to as **conformance checking**) [RVdA08, CvDSW18, BC17], and to improve the processes by processing the information of previously executed process instances (referred to as **process enhancement**) [RvdA06, VdASS11]. While current process mining algorithms are providing useful insights, three major drawbacks can be observed. First, the results are often generated using information of process execution logs, which are created after a process instances have been executed (referred to as **offline process mining**). Second, while the focus on the execution of a business process enables companies to acquire knowledge of the actual execution of business process and thus supports quicker changes in the business process logic, other perspectives of a business process are oftentimes neglected, i.e., data elements attached to events or even data points outside of the business process, e.g., periodic measurements of vital signs of a patient in the ICU of an hospital. Third, according to [The19] “*[...] world-class organizations leverage business process change as a means to improve performance, reduce costs, and increase profitability*”. Thus deviations in the behavior of the executed process and the designed process need to be detected as early as possible and quantified to calculate the expected cost.

This thesis aims at overcoming these drawbacks by developing **online process mining** algorithms, which can be applied directly on a stream of events while a process is still active and by taking different perspectives into account, i.e., data elements of events and external data points, for process discovery and process enhancement.

1.1 Motivation

Process Mining [VDAADM⁺11], established itself as valuable asset for companies [KSSI20]. Gartner [KSSI20] says, “*[...] digital transformation drives growth in business users’ awareness of the benefits of analyzing and understanding their own processes and business operations in a broader enterprise context*”, highlighting the importance of creating a shared business understanding of processes, to make them more efficient. A process model describes the logic of a business process, i.e., which activity is followed by which activity, which activities are executed simultaneously, which activity prevents the execution of

another activity and so on. For the visualization of a process model, different notations are available, including Petri Nets [Pet81], Business Process Modeling Notation (BPMN) [OMG13], and Unified Modeling Language (UML) [Fow04]. An example of a process model modeled in BPMN can be seen in Figure 1.1a.

The execution of a process model is realized by a process instance. A process instance contains information of a process for one specific case. Figure 1.1b shows an example of attribute values for an instance, i.e., the name of the patient, Jane Doe, as well as her age and the level of severity of the current suffered symptoms. These values are collected by a health provider.

The process model describes the guidelines for treating gall stones and have been gathered from the official website of the Austrian health ministry¹. In the beginning of the process, the severity of a patient's symptoms are evaluated from the instance attributes (cf. Figure 1.1b). If a patient is suffering a moderate level of symptoms due to gall stones, antispasmodic drugs and abrosia, i.e., the abstinence of food, are administered. Note that the medication and the abstinence of food are to be executed in parallel, i.e., the order of both activities is not determined and succeeding activities can only be executed after both activities finished their execution. In addition to these events, a pain therapy can be performed if necessary. Since a possible pain therapy is performed in parallel, it has to be finished as well, before succeeding events can be executed.

If the severity of symptoms is at a severe level, a surgical removal of the gall bladder is planned, a cholecystectomy. For this, the patient has to be admitted to a hospital. Afterwards the cholecystectomy is performed and the patient is dismissed in the end.

Such process models are designed by domain experts but the process instances are executed by different parties. Therefore the execution does not always match the designed process model [CW99].

Process mining aims to mitigate the differences between the described behavior and the execution and analyzes the executed behavior of processes. The different algorithms of process mining are categorized into the following three categories [VDAADM⁺11]too:

- **Process discovery** algorithms use the logged data of process instances of a business process and mine a process model which corresponds to the logged data.
- **Conformance checking** algorithms monitor the behavior of logged data of process instances related to a process model for this business process, detect deviations and calculate the fitness of the logged data.
- **Process Enhancement** algorithms use logged data of process instances as well as a process model to detect bottlenecks in a process model and aim at improving the business process, i.e., suggesting a different process model after a change the in logic of the process has been detected.

The logged data of process instances is stored in a process execution log (cf. Table 1.2). A log contains several process instances, i.e., traces. Each of these traces can contain different information on a process instance (cf. Figure 1.1b). A trace consists of several events. An event comprises the actual information of an executed activity, e.g., which activity has been executed, by whom, in which state is the activity now, and when has this

¹<https://www.gesundheit.gv.at/krankheiten/verdaunung/gallenblase/gallensteine-therapie>

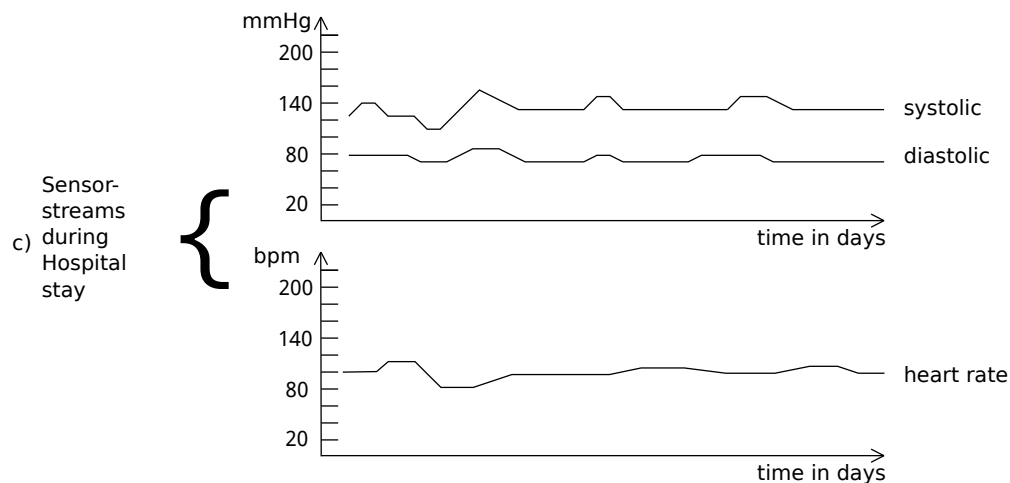
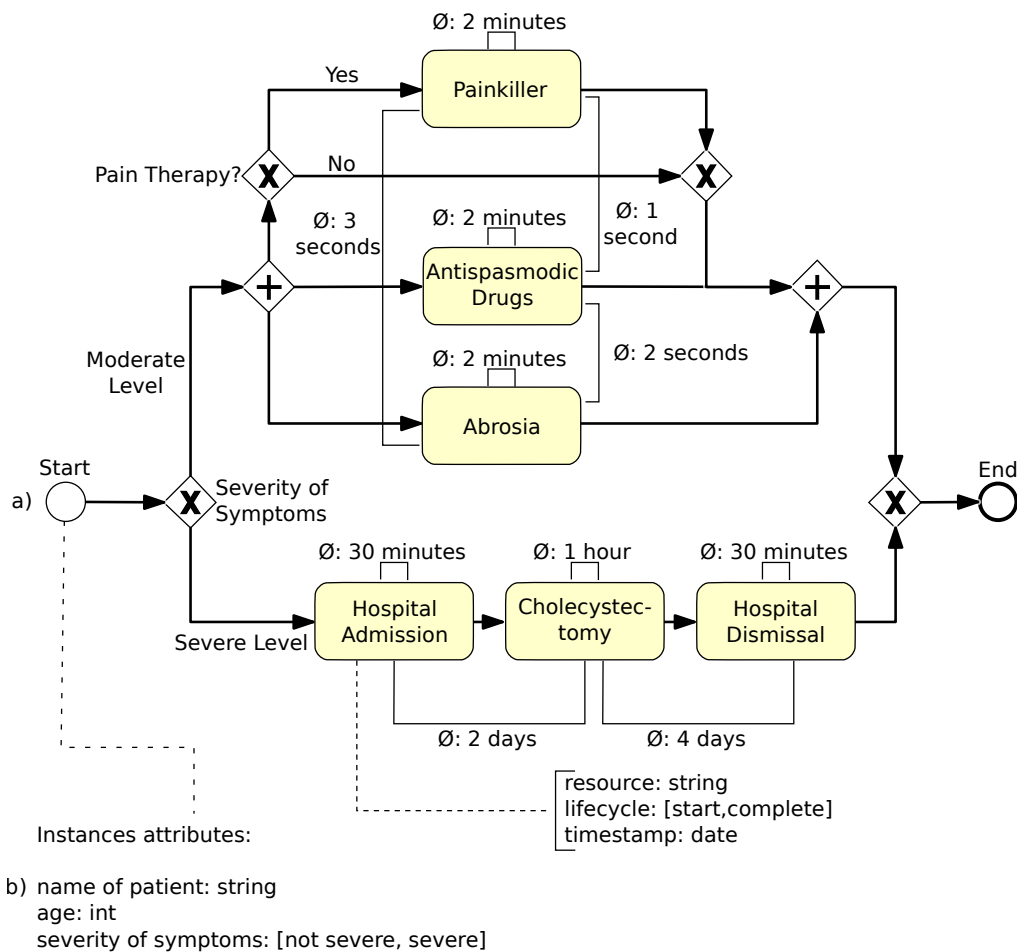


Figure 1.1: Example of a process model describing the treatment for gall stones designed after description from <https://www.gesundheit.gv.at/krankheiten/verdaueung/gallenblase/gallensteine-therapie>

Table 1.1: Example of data elements on the instance level for Table 1.2

Trace information			
trace	name of patient	age	severity of symptoms
1	Jane Doe	20	Severe
2	Gustav Schmidt	33	Moderate
3	Lisa Eipel	31	Moderate

been done. An example of data elements attached to an event can be seen in Figure 1.1a at activity “Hospital Admission”, i.e., the performing resource, i.e., a staff worker of the hospital, the current lifecycle status and the point in time of the execution. The lifecycle of an activity reflects the state of an activity, i.e., if the activity is starting, complete, or in any other state. The number of data elements attached to an event is unlimited. In addition to data elements, an activity can carry certain temporal properties. Every activity has an execution time, i.e., the time between the start event and the end event of an activity, e.g., 30 minutes for the “Hospital Admission”. Furthermore the time gap between two activities can be expressed as well, i.e., the time between the end event of one activity and the start event of the next activity, e.g., 4 days between “Cholecystectomy” and “Hospital Dismissal” (cf. Figure 1.1a).

An example of a process execution log can be seen in Table 1.2 with the corresponding process instance information in Table 1.1. Each row in Table 1.2 represents an event for a process instance of the business process seen in Figure 1.1a. The first column displays the trace id for a logged event. The second column contains the name of an event. In this process execution log the acting resource of an activity, the lifecycle status and the timestamp of the execution are logged in the following columns. As can be seen in Table 1.1, the patient of trace 1 suffers a severe level of symptoms, whereas the other patients only suffer a moderate level of symptoms. Hence the events “Hospital Admission”, “Cholecystectomy”, and “Hospital Dismissal” are present for trace 1 and not in traces 2 and 3. Both follow the other path depicted in the process model. Note that, since the process model has the activities “Abrosia”, “Antispasmodic drugs” and optionally “Painkiller” depicted in parallel, the order of these events can vary. In trace 2 “Antispasmodic drugs” is executed before “Abrosia” where the order of these events is switched in trace 3 with the event “Painkiller” preceding them as well.

1.2 Problem Statement and Challenges

In traditional process mining, the control flow of a process is analyzed using the information of a process execution log. These log files are generated after the process is finished, i.e., every event generated by the actors in a process are collected and put into a log. Thus the analytical results of a process are retrieved ex-post, i.e., based on offline process mining. To compensate this drawback, event streams [IEE16] instead of process execution logs can be used as an input for process mining algorithms, while the processes are still being executed, i.e., using online process mining.

What both variants of process mining have in common, is the focus on the control flow of a process, but more information is available in events that influences the process logic and control flow of a process. Most of the data elements that are available in events

Table 1.2: Example of process execution log for the example from Figure 1.1. Each id is matching an id from Table 1.1.

Process Execution Log				
trace	event	resource	lifecycle	timestamp
1	Hospital Admission	John Doe	start	20-05-2021 14:30:24
1	Hospital Admission	John Doe	complete	20-05-2021 14:50:14
1	Cholecystectomy	Jane Shepard	start	22-05-2021 11:15:20
1	Cholecystectomy	Jane Shepard	complete	22-05-2021 12:18:10
1	Hospital Dismissal	Steve Smith	start	26-05-2021 09:15:33
1	Hospital Dismissal	Steve Smith	complete	26-05-2021 09:42:16
2	Antispasmodic drugs	Joanne Miller	start	24-05-2021 09:23:10
2	Antispasmodic drugs	Joanne Miller	complete	24-05-2021 09:24:10
2	Abrosia	Joanne Miller	start	24-05-2021 09:24:11
2	Abrosia	Joanne Miller	complete	24-05-2021 09:26:06
3	Painkiller	Joanne Miller	start	25-05-2021 09:23:11
3	Painkiller	Joanne Miller	complete	25-05-2021 09:25:16
3	Abrosia	Joanne Miller	start	25-05-2021 09:25:17
3	Abrosia	Joanne Miller	complete	25-05-2021 09:26:00
3	Antispasmodic drugs	Joanne Miller	start	25-05-2021 09:26:10
3	Antispasmodic drugs	Joanne Miller	complete	25-05-2021 09:28:11

are not used by existing approaches, even though they could potentially provide useful insights. Recent literature emphasizes the need to retrieve information on processes as soon as possible, [KSSI20, Rei20, CLS20], as well as, to develop process mining algorithms to process the ever increasing amount of data [vZvDvdA18, BC17, vZBH⁺17]. To meet this demand and to incorporate data elements into process mining algorithms, several problems and challenges arise.

The first problem, is concerning the point of time process mining algorithms which use data elements, are applied. To enable online process mining algorithms, i.e., algorithms applied during the execution of processes, events have to be created immediately and collected in a suitable data set for process mining algorithms. A process execution log can be seen in Table 1.2. Even though tasks executed by non-human resources can create events immediately, i.e. the information system in a hospital for the activity “Hospital Admission”, human resources tend to create events when time is available [AZAMBH18]. Thus the execution log is generated afterwards, i.e., for the activity “Painkiller”. Therefore a solution is required, that supports human resources to create events immediately without adding an additional workload for the human resources. Another important factor is the quality of the event information, i.e., if a human resource is executing an activity, the event information can vary between different human resources, e.g., while one nurse is capturing additional information of a patient, another nurse may only be capturing the point of time of the execution. Thus it is important to support human resources even further and generate events systematically providing the same level of documentation, to produce a stream or a log of events to apply process mining algorithms on.

In addition, working with a stream of data, poses stream specific problems [Agg07], e.g., even though the data points in the stream often contain a small amount of data elements,

the amount of data is infinite and therefore cannot be stored and must be processed immediately (cf. Figure 1.2 (I)).

Another problem are **concept drifts** in processes [BVDAZP14, MBCS13, WK96]. Processes evolve and are changed by domain experts. Every time a change is committed to the logic of a business process, e.g., a new activity is introduced, an activity is removed, or the order of activities got changed, the change is not always made subsequently in the current process model. This mismatch between the newly designed process logic by domain experts and the currently used process model can be leading to unexpected results of process mining algorithms. Deviations in the behavior of process instances compared to the desired behavior defined in a process model are calculated using conformance checking algorithms [CvDSW18, MdLRvdA16a, VdAAvD12]. For example, a deviating process instance with an additional event could be detected and determined as unfit, where in reality the added event has been added by domain experts and the process model has not been updated just yet. Incorrectly detected deviations can be costly, thus process mining algorithms must be able to deal with a new process logic fast, e.g. the General Data Protection Regulation (GDPR)² legislature required processes to be adapted and deviations to the new legislature could cost up to “*20 million euros [...] or 4% of the total worldwide annual turnover of the preceding financial year*”, according to [Dro17]. While there is already work existing for performing conformance checking online [BC17, vZBH⁺17], only the control flow perspective is usually taken into account. Current literature shows that the data elements in a process can reflect the behavior of process instance without requiring the actual sequence of events in a process instance [RTG⁺20], thus motivating the need for new conformance checking algorithms which take data elements in the workflow of a process into account.

Additionally, concept drifts can be detected and categorized it into a specific concept drift [BVDAZP14]. The literature describes four different kinds of drifts. A recurring drift, reflects a seasonal pattern, i.e., the process model changes between two different models each summer and winter. An incremental drift reflects a small adaption in the process, i.e., a new activity added to the process model. The last two type of drifts are sudden drifts and gradual drifts. A sudden drift is happening if all process models immediately match the behavior of the new process model, while in gradual drift, the new process instances match the new behavior and the old instances match match the previous behavior. These drifts are described for changes in the control flow perspective of a process. As can be seen in Table 1.2, events contain more information than only control flow data, therefore not only can the control flow of a process change, but also the logic rules for data elements of a process can vary [RW12], i.e., a legislature change could allow pharmacists to administer painkiller drugs instead of only medical doctors or a new medical study enforces a longer stay period for patients in the hospital after a cholecystectomy, leading to an increased temporal distance between the “Cholecystectomy” event and the “Hospital Dismissal” to 5 days instead of 4.

As mentioned before, concept drifts, a change in the logic of a business process, imposes several challenges for process mining algorithms. Even though different categories of concept drifts are already described in the literature [BVDAZP14], a formal definition to distinguish these drifts is lacking. In an online setting new challenges arise as well for concept drift detection. Working with an event stream instead of a process execution log,

²https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu_en

implies that there is an unlimited number of process instances. In these process instances the process logic could change at any point, thus process mining algorithms need to detect concept drifts fast. This also affects the detection of outliers in the data perspective. Each data element can require a different outlier detection, i.e., some numerical data elements can be normally distributed, while other data elements like the acting resource, follow a organizational diagram of a company (cf. Figure 1.2 (II)).

Another problem, concerning the monitoring of the behavior of process instances compared to the proposed behavior of a process model (cf. Figure 1.2 (III)), is the quantification of a deviating process instance. Deviations can be detected using conformance checking algorithms. These deviations are assigned a penalty cost using a cost function, but this function does usually not take the process history or data elements into account for adjusting the costs. By taking the control flow perspective into account as well as the data elements attached to the events, new cost functions should be able to adjust the severeness of a deviation for processes, e.g., could an overdose of a medical drug lead to major complications in a process for a patient in a hospital, while the missing of a medical drug could lead to only minor complications. This leads to new challenges, adjusting the deviation cost a process instance depending on the data elements of various events during the execution of a process instance.

In addition to data elements attached to events in a log or event stream, there are potential data streams outside of the process environment, i.e., the room temperature of a manufacturing domain, which could potentially influence a process. These external data sources, can help identify the reason for a concept drift and explain it. Even though some concept drifts are ultimately committed by domain experts, the reason for a change is often not visible in the log or event stream. While specific information on an event is captured in an event stream or log, an external sensor, e.g., a periodical blood pressure measurement, is not attached to a specific event, but a suspicious blood pressure could explain certain deviations, e.g., a planned medical drug could be prohibit by a specific blood pressure, which can lead to a drift. A challenge hereby is the integration of an external sensor data stream into a log or event stream. As can be seen in Figure 1.1c, the blood pressure and heart rate of a patient are constantly measured during the whole stay of a patient in a hospital, i.e., from “Hospital Admission” to “Hospital Dismissal”. There is not a specific activity assigned to it, since such data is usually monitored over a period of time instead of one specific point in time. Thus the data preparation for generating a suitable data set for process mining algorithms need to be extended, to relate data point sequences of external sensor streams to a process instance. Changes in the behavior of the external data can help domain experts determine the root cause of a concept drift. Additional challenges evolve around the detection of relevant sensor streams for a process and the comparison of data streams, since traditional process mining algorithms are not suitable for comparing an arbitrary number of data points, i.e., measurements.

Finally, the solution for these problems, as well as the effects of the solutions need to be evaluated (cf. Figure 1.2 (IV)). While a generalization to other domains is a difficult task, an evaluation of elaborated algorithms and of process mining in general is required.

Figure 1.2 gives an overview of the previously identified research goals and their challenges. In the following the section the corresponding research questions of the previously described challenges and problems are defined.

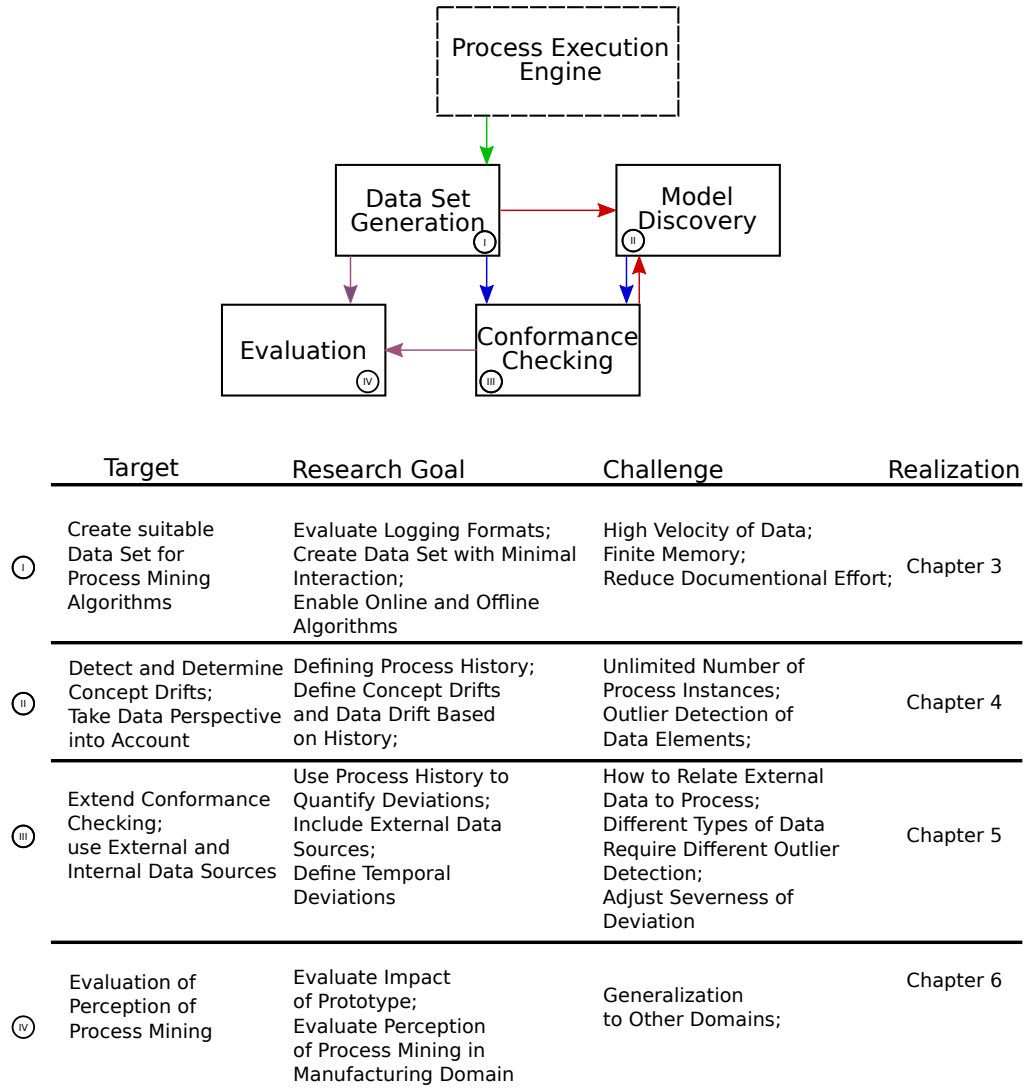


Figure 1.2: Overview of Targets, Research Goals, Problems and where addressed in this thesis.

1.3 Research Questions

In this section, the research questions of this thesis are stated and elaborated. Section 1.4 gives an overview, on how these research questions are tackled.

The first research questions discuss the technical properties for a suitable data set on which process mining algorithms can be applied on.

RQ ①a) How should a data format for process mining algorithms be designed to be created during the execution? While the human-readable structured XML serialization of XES provides a useful representation to be used in process execution logs, it still poses some liabilities in an online setting, where speed and data space are oftentimes limited. The tree structure of XML documents can be tedious to modify, since modifications have to be made at the correct position in a document, therefore the whole document has to be parsed, using memory and time. Desired properties for a suitable data set are

discussed and a serialization format is proposed for process execution logs to parse and extend them easily during and after the execution of a process.

RQ (1b) How to generate data sets directly at the execution of a process for online process mining algorithms? While a process execution log is generated after the process instances have been executed, an event stream is a data source containing events, which are generated directly when they are executed. An approach to generate an event directly when it is executed, containing all necessary and desired data elements for process mining algorithms is elaborated in connection with a process execution engine.

RQ (1c) How to improve the quality of the data set and reduce the error-proneness?

Tasks in business processes oftentimes are executed by one or more human actors. The documentation of a task in a business process is typically not a desirable duty for human actors and a time consuming task. Thus the documentation is often made when the human actor finished all other tasks at the end of their shift and is prone to errors due to the time gap between the execution of the activity and the documentation task. To support human actors by reducing the amount of time spent on the documentation of their activities as well as increasing the quality of the documentation, different technologies, are investigated and combined with a process execution engine.

The following research questions discuss the evolution of a business process by taken multiple perspectives into account, i.e., a collection of process models and categorized concept drifts to document the history of a business process. The influence of data elements on concept drifts is discussed as well.

RQ (2a) How can the evolution of a business process be discovered at runtime?

A business process has to adapt constantly. The reasons can be manifold, i.e., a seasonal effect or a legislation change. While other approaches already looked into the analysis of concept drifts in a process execution log, a definition of the evolution of a process is missing. A formalization of the different types of concept drifts related to the evolution of a business process is elaborated as well.

RQ (2b) How do data elements relate to concept drifts? The main focus for a plethora of process mining algorithms is on the control flow of a business process, but events in a process execution log oftentimes contain a variety of data elements attached to them. In Figure 1.1a, data elements are shown for the activity “Hospital Admission”. For example, to optimize the workflow, patients could be admitted directly in the correct hospital unit instead of a general admission but has to be done by the medical personnel instead of the general personnel. This would lead to human actors of a different hierarchy level in the hospital organization, which is reflected in the event stream. This drift in the process logic is not reflected in the control flow of a business process but in the data perspective, a data drifts.

The following research questions discuss the conformance of single process instances, how deviations can be quantified and if the deviations can be explained.

RQ (3a) How to better quantify the costs of deviations between process instances

and a process model using data elements? Conformance checking tries to align the actual execution of a process instance to a process model. If there are deviations present, they are assigned a cost to them. Usually a default cost function, that assigns a fixed value to mismatch is used, but a deviation in the process mining data set does not necessarily reflect an error in the execution of a process, but could also reflect an error in the logging of the process instance. With the information of previously executed process instances in a process history, an amount of acceptable values for attached data elements can be

gathered and an advanced cost function be introduced.

RQ ③b) **How can the temporal perspective be taken into account for conformance checking?** While a deviation of a executed process instance compared to a process model can be detected by aligning both sequences, the tasks can still be executed incorrectly, e.g., the time passed between events or the duration of a task can exceed certain thresholds, i.e., taking too long or not long enough. The example in Figure 1.1a shows the average duration for each task and the average time between tasks. New conformance checking algorithms are necessary to take temporal deviations into account.

RQ ③c) **How to discover the source of a concept drift?** Concept drifts can be detected using process execution logs or event streams, but they are often not sufficient to detect the reason a concept drift happened, i.e., a recurring drift often reflects a seasonal change, so the temperature could be a potential cause for a drift. To analyze external sources, e.g., the heart rate and blood pressure in Figure 1.1c which are not included in an event stream or process execution log, external sensor data streams can be related to business process. As the source and explanation of a concept drift, a drift in the behavior of external sensor data streams at the same time can be identified.

RQ ④ **What are the general expectations on process mining algorithms by domain experts and what are the actual results after it has been introduced?** A plethora of process mining algorithms are already present and this thesis provides some new approaches to close present research gaps. The impact of process mining in general and of the elaborated algorithms in this thesis need to be evaluated by domain experts.

1.4 Methodological Background and Contributions

This thesis follows the **Design Science Research** approach. In [Wie14], design science is described as follows: “*In design science, we iterate over two activities: designing an artifact the improves something for stakeholders and empirically investigating the performance of an artifact in a context*”. In this research method the focus is on the artifact, i.e., it is artifact driven instead of problem driven with the main objective to draw prescriptive knowledge about the design of artifacts [vBM19]. Such artifacts range from metrics, formal definitions, complete frameworks to algorithms.

The contribution of this thesis is structured into artifacts, containing algorithms to tackle the before mentioned challenges and provide new meaningful insights in processes. The artifacts together form the main artifact of this thesis, an online process mining framework providing online process mining algorithms as well as a method to generate data sets suitable for process mining, called TIDATE.

Using the six core dimensions to design science research, according to [vBM19], this thesis can be categorized as follows:

- **Problem Description:** Deviations in business processes oftentimes are detected after the processes have been executed. Information is more valuable if gathered earlier.
- **Input Knowledge:** Consisting of interviews with domain experts, real-world process information of the manufacturing domain, cloud-based process execution engine, prior knowledge from previous process executions and external data streams from sensors

- **Research Process:** Creating artifacts in form of algorithms and prototypes. Evaluated on real-life process data as well as through domain experts using focus groups [Kru14] and technical action research [WM12].
- **Key Concepts:** Creating data sets suitable for process mining as well as supporting human actors working with a process execution engine. Discovering the evolution of a business process. Also to detect, quantify and explain deviations in process instances and discover opinions of process mining of domain experts through focus groups.
- **Solution description:** Extending process execution engine to generate a data set for process mining while processes are being executed, enabling online and offline process mining algorithms. Collect the evolution of a business process by detecting concept drifts [BvdAŽP11] and analyze the differences between the process models as process history. Include data perspective for process models and use process history and external data streams to quantify deviations in process instances. Evaluate impact using focus group interviews.
- **Output Knowledge:** Process execution logs and event streams are generated. A process history containing identified and classified concept and data drifts is built. The alignments costs are adapted through the inclusion of process history and external data streams in conformance checking.

The artifacts of this thesis are now outlined per chapter.

Chapter 2, Related Work, provides the related work und fundamentals required to follow the contribution.

Chapter 3, Creating Data Sets for Process Mining, contains the methods for the creation of a suitable data set for process mining as well as a discussion on the format the data set is serialized in. The content of Chapter 3 is based on the following publications:

Stertz, F., Rinderle-Ma S., Hildebrandt T., Mangler J.: Testing Processes with Service Invocation: Advanced Logging in CPEE. In: 14th ICSOC Workshops (2016), Pages: 189-193
https://doi.org/10.1007/978-3-319-68136-8_22

Stertz, F., Mangler J., Rinderle-Ma S.,: Balancing Patient Care and Paperwork Automatic Task Enactment and Comprehensive Documentation in Treatment Processes. In: Enterprise Model. Inf. Syst. Archit. Int. J. Concept. Model. (EMISA) Vol. 15, 2020, Pages: 34-48

An overview of the contribution of this chapter can be seen in Table 1.3. This thesis is using the cloud-based process execution engine, CPEE [MRM14]. While tasks are being executed, the process execution engine is producing notifications to different services. This thesis provides a logging services, which generates an event stream and a process execution log, both following the XES format [IEE16], on which process mining algorithms can be applied on. Section 3.1 represents a new approach to serialize the XES format to suit the needs of a fast lightweight logging format which is human readable and can be used to

Table 1.3: Details of Chapter 3

Chapter 3	
Input	<ul style="list-style-type: none"> - Execution of Process Instance - Stream of non formatted events
Output	<ul style="list-style-type: none"> - Process Execution Log - Event Stream (XES) - Human readable Documentation
Artifacts	<ul style="list-style-type: none"> - New Serialization Format of XES - Extension of Process Execution Engine - Prototype using NFC tags with Process Execution Engine

apply process mining algorithms on. Furthermore the necessary properties of the format and its requirements are discussed (RQ (1a)). Section 3.2 describes the extensions of the process execution engine to generate a data set directly at the execution. The extension provides a process execution log directly after an event is detected and injects it into an event stream as well for further services (RQ (1b)). In Section 3.3, the quality of a process documentation is discussed with involved human actors. For this, use cases from the care domain are treated from a process oriented view in a nursing home. The staff members of a nursing home have to log each task they perform to due to legal legislation, but because of a heavy workload the documentation is often postponed to the end of their shift which can lead errors. A novel approach is presented using NFC³ technology and a process execution engine to support staff members of a nursing home, generate a data set directly at the execution and transform to a fitting documentation for the nursing home (RQ (1c)).

Chapter 4, Discovering the Evolution of Processes through Concept Drifts, focuses on online process discovery by defining and classifying concept and data drifts. This allows to collect a **process history**, which represents the evolution of business process. The content of Chapter 4 is based on the following publications:

Stertz, F., Rinderle-Ma S.: Process histories-detecting and representing concept drifts based on event streams. In: On the Move to Meaningful Internet Systems. OTM 2018 Conferences - Confederated International Conferences (CoopIS), Pages: 318-335
https://dx.doi.org/10.1007/978-3-030-02610-3_18

Stertz, F., Rinderle-Ma S.: Detecting and Identifying Data Drifts in Process Event Streams Based on Process Histories. In: CAiSE Forum 2019, Pages 240-252,
https://dx.doi.org/10.1007/978-3-030-21297-1_21

An overview of the contribution of this chapter can be seen in Table 1.4. With the results from Chapter 3, an event stream is now generated when tasks are being executed and processed in the process execution engine, CPEE. In this chapter of the thesis, changes in the executed process instances are detected to identify drifts in the process model.

³Near-field communication

Table 1.4: Details of Chapter 4

Chapter 4	
Input	<ul style="list-style-type: none"> - Process model with data elements - Event Stream
Output	<ul style="list-style-type: none"> - Process model with data elements - Process history
Artifacts	<ul style="list-style-type: none"> - Algorithms to generate a Process history - Data Drift formalization - Drift formalization related to process history

These drifts can be caused by different sources. A recurring drift, for example, represents seasonal changes for example, while an incremental drift reflects small changes may be due to a legislation change [BVDAZP14]. To classify the different types of concept drifts, a formalization of a process history and the definition of four concept drift types according to process history are presented in Section 4.1 (RQ 2a). This approach is prototypically implemented and evaluated through an artificial data set, inspired from an use case from [BVDAZP14]. It shall be noted, that these concept drifts are not only detectable on the workflow perspective, but also the data perspective can contain changes, i.e., data drifts. In Section 4.2, algorithms are elaborated to detect drifts in an event stream focusing on the data elements attached to events. All of these data drifts are classified and formalized (RQ 2b). The approach is prototypically implemented and evaluated through a real-world log from the manufacturing domain.

Chapter 5, Time & Data-Aware Conformance Checking and Explaining Drifts, focuses on conformance checking, detecting the deviations in process instances and trying to explain the deviations. While oftentimes, conformance checking is only used with a standard cost function, a more refined cost function is elaborated and external sensors are used to explain deviations in the process instances. The content of Chapter 5 is based on the following publications:

Stertz, F., Mangler J., Rinderle-Ma S.: Analyzing Process Concept Drifts Based on Sensor Event Streams During Runtime In: 18th Business Process Management, BPM 2020), Pages: 202–219
https://doi.org/10.1007/978-3-030-58666-9_12

Stertz, F., Mangler J., Rinderle-Ma S.: Data-driven Improvement of Online Conformance Checking. In: International Enterprise Distributed Object Computing Conference, EDOC 2020), Pages: 187-196
<https://doi.org/10.1109/EDOC49727.2020.00031>

Table 1.5: Details of Chapter 5

Chapter 5	
Input	<ul style="list-style-type: none"> - Process model with data elements - Event Stream - External Sensor Streams
Output	<ul style="list-style-type: none"> - Alignments - Deviation costs - Process Model infused with Time Series - Temporal Profiles
Artifacts	<ul style="list-style-type: none"> - Advanced Cost Function for Conformance Checking - Algorithms incorporating external Sensor - Temporal Profiles - Algorithm to detect reason for concept drift

Stertz, F., Mangler J., Rinderle-Ma S.: Temporal Conformance Checking at Runtime based on Time-infused Process Models. Tech. rep. (2020), <https://arxiv.org/abs/2008.07262>

An overview of the contributions of this chapter can be seen in Table 1.5. With the results from Chapter 3 and Chapter 4, a process model and an event stream is now used as input to detect and analyze deviations of a process instance compared to the process model.

Section 5.1 introduces a more refined cost function for conformance checking in general. In comparison to the default cost function in conformance checking, where a deviation in the log and a deviation in the model are assigned a cost of 1, this approach adapts the cost of a deviation by tracking the data elements attached to events. Using the process history, acceptable values for data elements can be determined and the deviations in the data elements are used to adapt the cost of deviation in the process instance accordingly (RQ 3a).

In Section 5.2, temporal profiles are introduced, to detect the average time between tasks and the time for a task to complete. Using this temporal profiles, the temporal behavior of process instances is inspected. Using the **z-score** [CA86], temporal deviations are calculated and used to assign costs to a process instance using conformance checking (RQ 3b).

Section 5.3 presents a novel approach to detect deviations which is taking external sensors into account. These sensors produce data streams containing measurements, e.g., temperature of a room or measurement of a work piece. The data streams are interpreted as time series and are related to a process instance. The time series are compared using Dynamic Time Warping (DTW) [BC94] and every time a concept drift is detected in a process history, deviations in the time series are detected to explain the reason of the drift in the process history (RQ 3c).

Chapter 6, Evaluating TIDATE - Time and Data Aware Process Mining at Runtime, contains the evaluation of the algorithms. There are two different methods used for evaluating this thesis, Technical Action Research (TAR) [WM12] and Focus Groups

interviews [Kru14]. The content of Chapter 6 is based on the following publication:

Stertz, F., Mangler J., Scheibel B., Rinderle-Ma S.: Expectations vs. Experiences–Process Mining in Small and Medium Sized Manufacturing Companies
 In: International Conference on Business Process Management Forum, Pages: 195-211
https://www.doi.org/10.1007/978-3-030-85440-9_12

In Section 6.1 the TIDATE framework is validated using the TAR approach (RQ ④). Section 6.2, investigates the general perception of process mining of members of manufacturing companies, before and after there have been exposed to process mining in their areas. To accomplish this evaluation, focus group interviews have been conducted. The participants are categorized by two layers. The first layer differentiates on the hierarchical position of a participant in the company, while the second layer categorizes participants on their level of exposure to process mining in their area, i.e., is process mining already introduced or not.

Chapter 7, Conclusion gives a short summary of the contributions of the complete thesis and an outlook for future work.

1.5 Thesis Structure

This thesis presents new algorithms incorporating data elements attached to events for process mining, reflected in the TIDATE framework. These algorithms can be applied at any time during the execution of processes and provide useful insights in behavior of process instances. Additions, like the advanced cost function for conformance checking algorithms, are presented as well as new approaches to explain the source of a drift in a process.

The remainder of this thesis contains the following chapters:

- **Chapter 2, Related Work.** This chapter summarizes the related work for this thesis and explains the necessary fundamentals to follow this thesis..
- **Chapter 3, Creating Data Sets for Process Mining.** This chapter focuses on the generation of a suitable data set on which process mining algorithms can be applied on. This enfolds offline process mining algorithms, i.e., after the execution of a business process and online process mining algorithms, i.e., during the execution of a business process. To achieve this, a new serialization format is presented and an approach to create a process execution log and event stream during the execution of business process using TIDATE.
- **Chapter 4, Discovering the Evolution of Processes through Concept Drifts.** This chapter focuses on the creation of a **process history** reflecting the evolution of a business process. A process history contains all different process models that have been detected via TIDATE after a concept drift has been detected and identified. In addition to concept drifts, drifts in the data elements are analyzed as well and data drifts are introduced.

- Chapter 5, **Time & Data-Aware Conformance Checking and Explaining Drifts**. This chapter describes a novel approach to combine exterior data sources, e.g., sensor data streams or the history of previous business process executions, to better quantify a deviation in the behavior of process instance to the process model. In addition to this, exterior data sources can be used to identify the cause of a deviation in a process instance.
- Chapter 6, **Evaluating TIDATE - Time and Data Aware Process Mining at Runtime**. In this chapter, the TIDATE framework and the impact of process mining in general is being evaluated using the Technical Action Research (TAR) method [WM12] and focus group interviews [Kru14].
- Chapter 7, **Conclusion**. The last chapter gives a brief summary of all gathered results with a discussion and possible future work is outlined.

2 Related Work

In this chapter the related work is outlined as well as the fundamentals to follow the contribution of this thesis. The content is based on the following publications:

Stertz, F., Rinderle-Ma S., Hildebrandt T., Mangler J.: Testing Processes with Service Invocation: Advanced Logging in CPEE. In: 14th ICSOC Workshops (2016), Pages: 189-193
https://doi.org/10.1007/978-3-319-68136-8_22

Stertz, F., Mangler J., Rinderle-Ma S.: Balancing Patient Care and Paperwork Automatic Task Enactment and Comprehensive Documentation in Treatment Processes. In: Enterprise Model. Inf. Syst. Archit. Int. J. Concept. Model. (EMISA) Vol. 15, 2020, Pages: 34-48

Stertz, F., Rinderle-Ma S.: Process histories-detecting and representing concept drifts based on event streams. In: On the Move to Meaningful Internet Systems. OTM 2018 Conferences - Confederated International Conferences (CoopIS), Pages: 318-335
https://dx.doi.org/10.1007/978-3-030-02610-3_18

Stertz, F., Rinderle-Ma S.: Detecting and Identifying Data Drifts in Process Event Streams Based on Process Histories. In: CAiSE Forum 2019, Pages 240-252,
https://dx.doi.org/10.1007/978-3-030-21297-1_21

Stertz, F., Mangler J., Rinderle-Ma S.: Analyzing Process Concept Drifts Based on Sensor Event Streams During Runtime In: 18th Business Process Management, BPM 2020), Pages: 202=219
https://doi.org/10.1007/978-3-030-58666-9_12

Stertz, F., Mangler J., Rinderle-Ma S.: Data-driven Improvement of Online Conformance Checking. In: International Enterprise Distributed Object Computing Conference, EDOC 2020), Pages: 187-196
<https://doi.org/10.1109/EDOC49727.2020.00031>

Stertz, F., Mangler J., Rinderle-Ma S.: Temporal Conformance Checking at Runtime based on Time-infused Process Models. Tech. rep. (2020), <https://arxiv.org/abs/2008.07262>

Stertz, F., Mangler J., Scheibel B., Rinderle-Ma S.: Expectations vs. Experiences—Process Mining in Small and Medium Sized Manufacturing Companies
In: International Conference on Business Process Management Forum, Pages: 195-211
https://www.doi.org/10.1007/978-3-030-85440-9_12

Stertz, F., Mangler J., Rinderle-Ma S.: The Role of Time and Data: Online Conformance Checking in the Manufacturing Domain. Tech. rep. (2021), <https://arxiv.org/abs/2105.01454>

This thesis is placed in the field of process mining. Process mining aims to analyze event information of business processes by discovering workflows, monitoring the behavior of process instances, identifying bottle necks and improving process models [VDAADM⁺11].

Process mining consists of three different areas: process discovery, conformance checking and process enhancement [vdA16]. As a main input, a **process execution log** is used, often employing the eXtensible Event Stream (XES) format [IEE16]. A process execution log contains a sequence of **events**. Each event relates to an activity in the process. An event is also part of a **trace**, i.e., a specific process instance. The complete collection of traces with their events is called a **log**, i.e., one log for one process. To inject more information into these process execution logs, data elements can be attached to events, e.g., the timestamp of an event. Data elements attached to events are not mandatory and can be arbitrary, but there is a small number of data elements widely accepted by a plethora of process mining algorithms, e.g., the name, the timestamp or the current lifecycle of an event [vdA16]. An example of a process execution log can be seen in Table 1.2 which contains events for the process shown in Figure 1.1a. In the following, the difference between offline and online process mining is explained with a detailed explanation of the three process mining areas afterwards.

2.1 Input Sources for Process Mining

There are two common input sources for process mining algorithms currently established. For standard process mining algorithms, a process execution log is used, i.e., a file containing all event information required for the algorithms following the XES format [VBDA11]. A file is created after the execution of a process. One file represents one process containing one root node, called log. A log contains the information on several process instances, called traces. Each trace then contains the information of all executed activities, called events, like acting resource or timestamp of execution. The second input source follows the XES format as well, but is used for online process mining, i.e., during the execution of a process, an event stream. Contrary to a process execution log, an event stream contains an unlimited number of events which poses several challenges since memory is finite and computational time should be short to handle the next event.

Figure 2.1 shows a simplified version of the running example from Figure 1.1a, which is used to demonstrate the most common process mining algorithms.

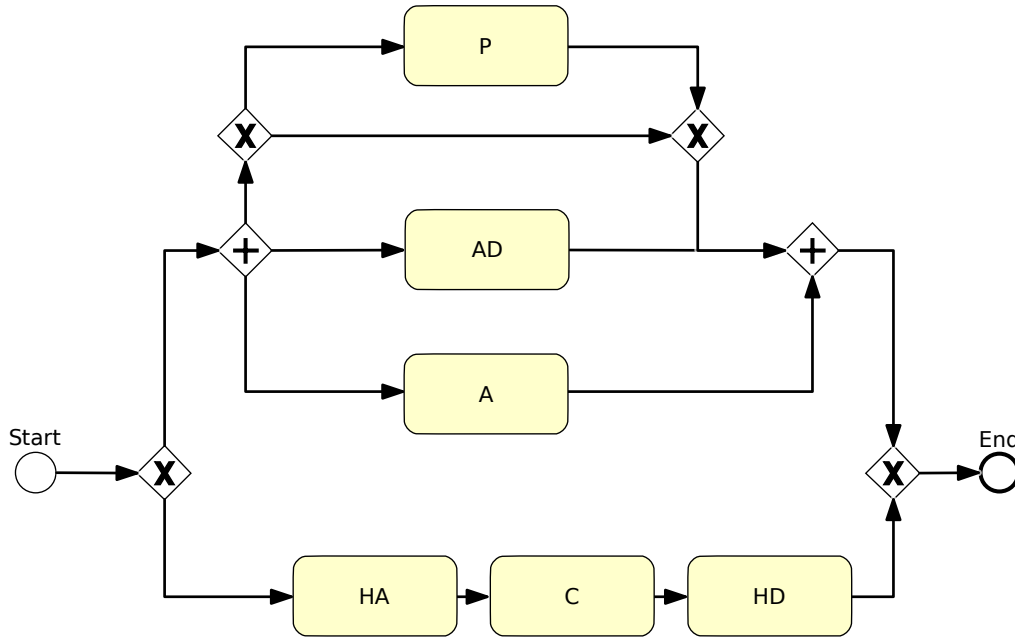


Figure 2.1: Process Model from Figure 1.1 with abbreviated labels for an easier understanding.

Figure 2.2 shows an example of a process execution log and an event stream, where both data snippets contain the same information. There are exactly 5 events on both data sets visible. On the left, in the process execution log, a complete log is present containing 3 events for the trace t_1 and 2 events for the trace t_2 . Both traces are fitting the process model (cf. Figure 1.1). The events are ordered by trace in a process execution log. On the right, an event stream is shown. In addition to the 5 events, events before HA are present in the stream as well as after HD . The events in the event stream are ordered by their timestamp, i.e., when their activities have been executed.

2.2 Process Discovery Algorithms

The α -miner is one of the first established algorithms [VdAWM04]. This mining algorithm searches a process execution log for specific patterns, i.e., which event is directly preceding or directly succeeding which event, which events are directly following each other at all and which events are not following any specific order. The algorithm defines the 4 following relations between two events, a and b from one process execution log.

- $a > b$ if in at least one trace b is directly following a .
- $a \rightarrow b$, if $a > b$ and $a \not\succ b$.
- $a \# b$, if $a \not\prec b$ and $b \not\prec a$.
- $a \parallel b$, if $a > b$ and $b > a$.

Process Execution Log Event Stream

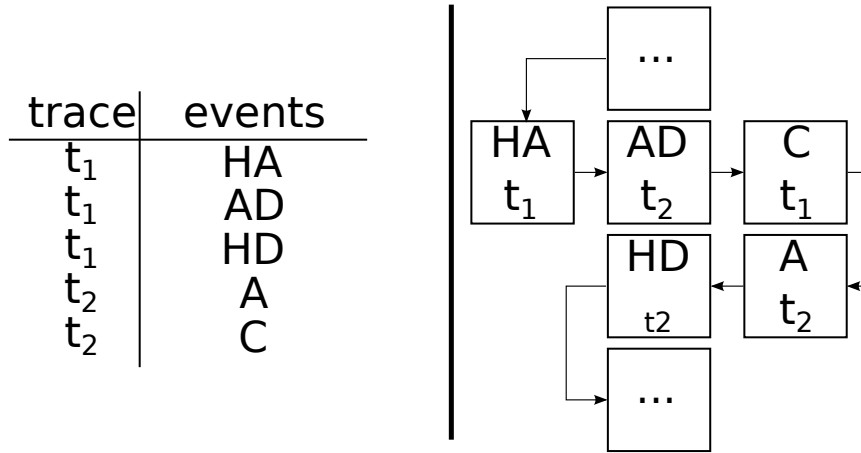


Figure 2.2: Example of process execution log and event stream for the process model in Figure 1.1

Using these definitions, specific workflow patterns can be discovered from a log, i.e., sequences, decisions and parallel patterns. A sequence pattern, for example $HA \rightarrow C$, is detected, since a “Cholecystectomy” is performed after the “Hospital Admission” and never before the “Hospital Admission”. A decision pattern, for example AD or HA after $Start$, is detected, since both events, AD and HA are executed directly after $Start$, i.e., $Start \rightarrow AD$ and $Start \rightarrow HA$, but $AD \# HA$ also applies, i.e., AD is not executed directly after HA nor HA after AD . This reflects the decision in the process model. Either “Hospital Admission” or “Antispasmodic Drugs” are executed in one process instance. The end of this decision is detected by looking at the following relations. $HD \rightarrow End$, $AD \rightarrow End$, and $AD \# HD$ represent the end of a split in a process model. Since AD , A , and P are executed in parallel, all of the decision patterns are detected for the other activities as well.

A parallel pattern is present in this process model, i.e., A , AD and P , where P is optional. This is detected, since $Start \rightarrow AD$, $Start \rightarrow A$, $Start \rightarrow P$, $AD \parallel A$, $AD \parallel P$ and $A \parallel P$ are present. A limitation of the α -algorithm is now visible, since $P \# AD$ or $P \# A$ cannot be detected. Therefore this decision pattern is lost using these relations. The join for the parallel pattern is detected, since $AD \rightarrow End$, $A \rightarrow End$, $P \rightarrow End$, $A \parallel P$, $AD \parallel P$, and $AD \parallel A$ are present.

The α -miner starts by scanning for every event in a log, the possible start events and the possible end events. Afterwards computationally intensive steps are applied to detect all possible sequences from one or more activities to others and filtered by removing **nonmaximal** pairs [VdAWM04]. These results are then finally processed into a workflow net.

Two major limitations of the α -miner are short loops and infrequent sequences of a process. A short loop can consist of only 2 activities, where for example the sequence $\langle a, b \rangle$ can be repeated n times. This leads to the relations $a > b$ and $b > a$, thus determined as $a \parallel b$, where in reality these 2 activities are never performed in parallel. A similar problem, can be witnessed by using the example from above, where the decision pattern starting P

cannot be detected. The second limitation, i.e., infrequent sequences, makes the α -miner error-prone to incorrectly recorded traces. If, for example, one trace out of 1000 traces features a wrong order of events, the α -miner algorithm is not taking the frequency of certain relations into account and treats each relation equally. Extensions published in [DMvDVdAW04, MWVdAvdB02], extend the α -miner to mitigate these limitations, by adding new relations for short loops and taken the frequency into account, by adding dependency/frequency tables.

The heuristics miner [WvdA03] takes the frequency of sequences into account and thus is better capable to deal with noisy logs, i.e., infrequent wrong paths due to an error in the execution or in the logging. The heuristics miner algorithm generates a dependency graph, i.e., a model displaying only connections with a sufficiently high dependency measure. The dependency measure between two events in a log, e.g. b following a , is calculated using the difference between the number of occurrences of $a > b$ and $b > a$ divided by the sum of occurrences plus 1. Let L be a log for the process model in Figure 2.1, consisting of the following sequences:

- $\langle \text{Start}, HA, C, HD, \text{End} \rangle$, 5 times
- $\langle \text{Start}, P, A, AD, \text{End} \rangle$, 1 time
- $\langle \text{Start}, A, AD, \text{End} \rangle$, 2 times
- $\langle \text{Start}, A, AD, P, \text{End} \rangle$, 5 times
- $\langle \text{Start}, AD, A, P, \text{End} \rangle$, 3 times

The dependency measures for L are calculated in Table 2.1. A number close to 1 likely reflects this sequence. A number close to -1 likely reflects this sequence in an inverse order, and a number close to 0 represents no dependency to each other.

The corresponding dependency graph for Table 2.1 can be seen in Figure 2.3. For the dependency graph two thresholds need to be defined, one for the absolute frequencies of an event sequence and one for the dependency relation. The graph in Figure 2.3 is created using only sequences, that have been observed at least 3 times and a have dependency relation of at least 0.42. As can be seen in the graph, there are some connections missing, e.g., between P and A , because it has been observed only once. The advantage of the heuristics miner is the speed of the algorithm, since the dependency relation can be calculated in linear time. A major disadvantage is the missing knowledge of parallel and decision gateways, since only the occurrences of sequences are taken into account.

The inductive miner [LFvdA13] requires a process execution log as input and guarantees a sound and fitting model. The models discovered by the inductive miner are typically represented as a process tree, i.e., a structured presentation, consisting of smaller subtrees which are connected through operators. As a base for calculating the relation of all the events of a process execution log, a directly follows graph is created, similar to the α -miner.

The following operators are defined in the inductive miner:

- \rightarrow , a sequence from the left subtree to the right subtree
- \wedge , parallel execution of all subtrees.
- \times , an exclusive choice between all subtrees.

Table 2.1: Dependency relation between Tasks for the dependency graph in Figure 2.3

	Start	A	AD	P	HA	C	HD	End
Start	$\frac{0}{1}$	$\frac{7}{8}$	$\frac{3}{4}$	$\frac{1}{2}$	$\frac{5}{6}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{0}{1}$
A	$\frac{-7}{8}$	$\frac{0}{1}$	$\frac{5}{12}$	$\frac{2}{5}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{0}{1}$
AD	$\frac{3}{4}$	$\frac{-5}{12}$	$\frac{0}{1}$	$\frac{5}{6}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{3}{4}$
P	$\frac{1}{2}$	$\frac{-2}{5}$	$\frac{-5}{6}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{8}{9}$
HA	$\frac{-5}{6}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{5}{6}$	$\frac{0}{1}$	$\frac{0}{1}$
C	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{-5}{6}$	$\frac{0}{1}$	$\frac{5}{6}$	$\frac{0}{1}$
HD	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{-5}{6}$	$\frac{0}{1}$	$\frac{5}{6}$
End	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{-3}{4}$	$\frac{-8}{9}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{-5}{6}$	$\frac{0}{1}$

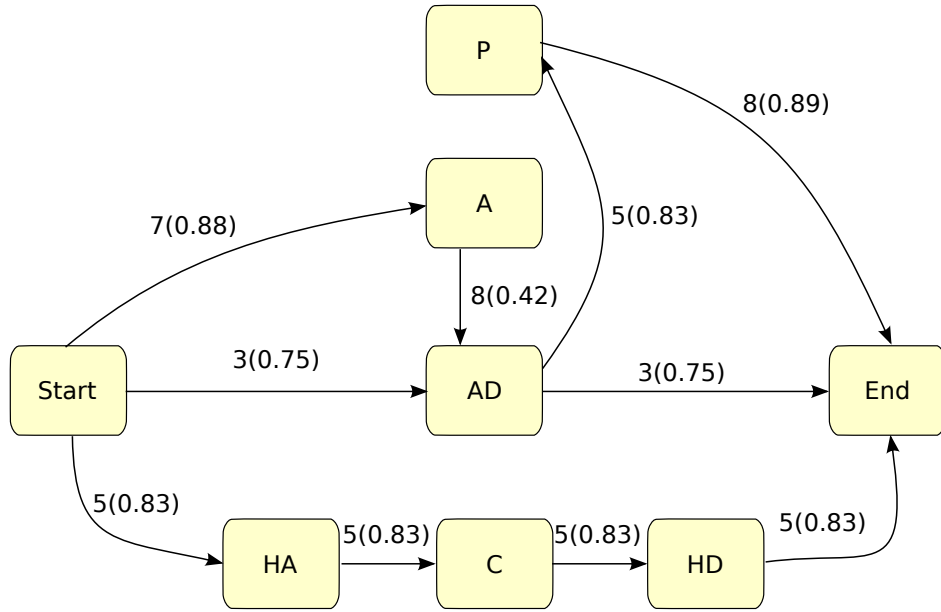


Figure 2.3: Example of dependency graph depicting the frequency of sequences and the dependency relation from Table 2.1

- \odot , a loop. The left subtree is the loop body while the right subtree is executed after the body, otherwise the loop ends.

The inductive miner is trying to find specific splits in a process execution log for each operator. The result of the process model from Figure 2.1 would be $\times(\wedge(\times(P,), AD, A), \rightarrow (HA, \rightarrow (C, HD)))$ Figure 2.4, shows where the splits are detected by the inductive miner for the model depicted in Figure 2.1, where the rectangles reflect splits within another the split of the outer rectangle.

A disadvantage of the inductive miner is, that it can only discover process models which can be described by process trees, since the underlying algorithm is generating process trees. Another limitation shared with other mining algorithms, is the requirement of a

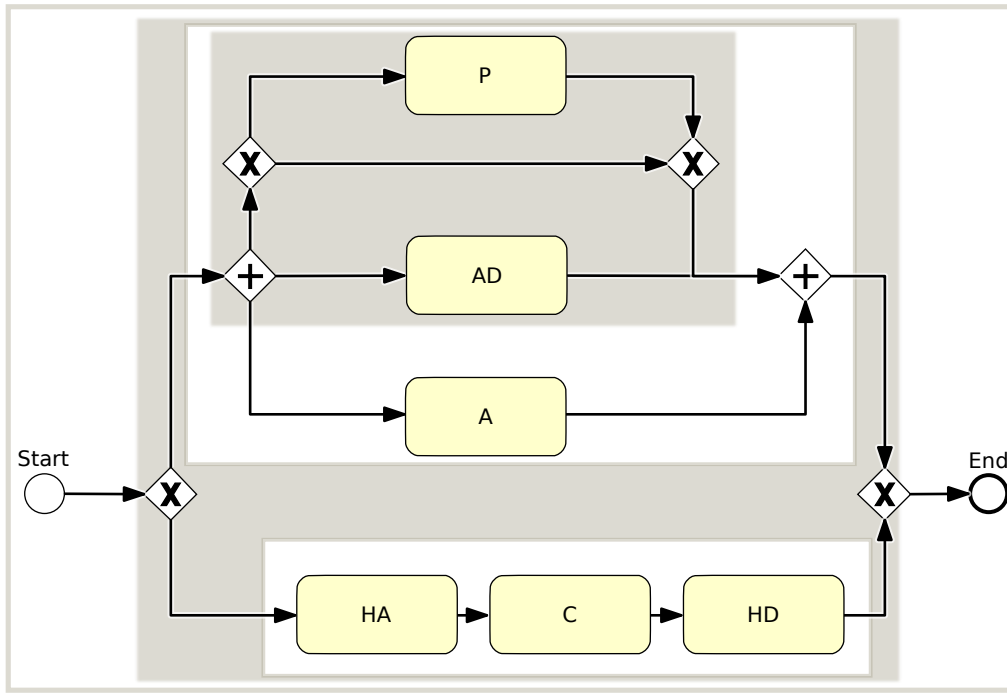


Figure 2.4: Example of the splits for the model in Figure 2.1. Following a similar design to depict the splits as seen in [LFvdA13]

sufficient process execution log, i.e., infrequent sequences that are not detected in a log are not present in the process tree and sequences that reflect a rare noise occasion are present in the process tree, which requires some processing of the process execution log before the algorithm can be applied..

Even though these are the three most common process discovering algorithms, there is a plethora of additional algorithms available, dealing with the limitations mentioned above, for example the genetic miner [vdAdMW05]. The genetic miner is capable to deal with noise and incomplete data in process execution logs, introducing a **Causal Matrix**, displaying the causal relation between activities, i.e., detecting the possible input and output activities for a specific activity.

2.3 Conformance Checking Algorithms

Several algorithms for process conformance checking exist [vdA16]. The first conformance checking algorithms, analyzed the behavior of process instances compared to a process model using token-replay in a Petri Net. After the replay the fitness of a process instance is then calculated using the missing and remaining tokens that still remain in the Petri net [RVdA08, VdAAvD12]. Nowadays alignments [Adr14] are used to calculate the fitness of a process instance by assigning a cost to missing or additional events in the event log and or the process model, called moves in the following. The different types of moves in an alignment are explained in detail using the notation from [CvDSW18] and the process model from Figure 1.1.

Synchronous moves are detected when the expected event in the trace and the event in

Table 2.2: Example of alignment with only synchronous moves.

<i>trace</i> ₁	Hospital Admission	Cholecystectomy	Hospital Dismissal
sequence	Hospital Admission	Cholecystectomy	Hospital Dismissal

Table 2.3: Example of alignment with a model move.

<i>trace</i> ₂	Hospital Admission	>>	Hospital Dismissal
sequence	Hospital Admission	Cholecystectomy	Hospital Dismissal

the model sequence are equivalent. For a synchronous move, usually no costs are assigned and a trace is fitting a process model if only synchronous moves are detected. A trace with three synchronous moves can be seen in Table 2.2, where all events from the trace match a possible execution of the process model.

Model moves represent that an event should be executed according to the process model at this position, but is not present in the event sequence of the trace. As can be seen in Table 2.3, the event “Cholecystectomy” is missing in *trace*₂. This model move is presented with a >> in the trace sequence.

If an event is appearing in the event sequence of a trace, but is not planned at this position in the process model, a log move is detected. In Table 2.4, the event “Hospital Admission” appears twice in the event sequence of *trace*₃, but is only expected once in the process model. Therefore, the symbol >> is placed in the trace row as annotation of a log move.

Calculating these alignments is an immense computational task, since lots of variations have to be considered finding the best alignment. To tackle this problem, an approximation of the conformance value is introduced by [SvZvdA20], which uses only a subset of all the possible sequences of the process model and approximates the alignments using the edit distance function. To identify events from each other in a process execution log or event stream, an identifier is required, e.g., the name of an event and the id of the related trace for example, which can be a source for errors [RMRJ11].

It is important to note, that current conformance checking techniques are usually not taking the likelihood of observed events into consideration. **Stochastic Conformance Checking** [LSvdA19] tackles this question. It translates event logs into a stochastic language, assigning a probability to each observed event sequence in a process execution log, and uses it to construct stochastic petri nets. Stochastic conformance checking then calculates the conformance of an event log by analyzing the distribution of the stochastic petri net and the event log using the Earth Movers’ Distance. This distance represents the cost for transforming a pile of earth, i.e. a distribution, to another one.

Another limitation of conformance checking approaches currently is the main focus on the control flow of a process instance without considering the data elements attached to the events in an instance, e.g., an activity is performed by a none authorized resource. Thus

Table 2.4: Example of alignment with a log move.

<i>trace</i> ₃	Hospital Admission	Hospital Admission	Cholecystectomy	Hospital Dismissal
sequence	Hospital Admission	>>	Cholecystectomy	Hospital Dismissal

the event is in the log, even though the activity has not been executed by a designated resource.

To overcome this, [RTG⁺20], do not use event information at all, but monitor the data elements related to a process instance in a data stream. The change of data elements are logged as a sequence of measurements, i.e., a time series. A workflow net is then enriched by these time sequences. The approach aims at the differences between the workflow net and a log of time sequences. In comparison to existing work, the event information is not used anymore. The differences between the time sequences is also not calculated on the time sequence as a whole, like the difference between two time sequences, i.e., using Dynamic Time Warping [BC94]. Instead the approach detects if the data elements are within certain intervals and increase or decrease at the expected point of time.

Recent approaches aim to perform conformance checking at run-time using an event stream [BSvdA14] instead of an event log. In such an online approach [vZBH⁺17], prefix alignments are calculated incrementally. This is necessary since traces in an event stream are not always complete all the time, i.e., a process instance can still be active and therefore some events are not present already in the event stream, but appear when the activities are executed. An optimal alignment would increase the costs, since the events are not present at this point in time. Prefix alignments allow to adjust the costs and use only present events, e.g., it does not increase the costs for an alignment if an event can still potentially appear in the event stream.

2.4 Process Enhancement

Process enhancement extends a process model, e.g., by introducing multiple perspectives [vdA11].

Organizational mining, for example, discovers patterns in the organizational perspective of a process execution log, i.e., the role of a resource or the executed tasks by one resource compared to another [SVdA08]. This information allows the detection of social networks from a process execution log as well [VDARS05]. A social networks describes how strongly connected two resources are to each other, as can be seen in Figure 2.5. The nodes represent the different staff members of a hospital, where a big node represents a resource that is often executing tasks, while a small node represents a resource that is executing only a few tasks from the process execution log, e.g., Dr. Shepard performs more tasks than the two staff members Doe and Smith. A thick arc represents a strong connection, i.e., these two resources work often in the same process instance, while a thin arc represents a loose connection, e.g., Doe executes activities often before Shepard and rarely afterwards, where Smith executes activities more often after Shepard. The time perspective focuses on the frequency of certain events and the timestamps of events, if available [BGvdW09]. They are used to detect certain bottlenecks to increase the efficiency of a process. Decision mining focuses on discovering rules for choices in a process based on the data elements of events in a process execution log [MDLRVDA16b].

Another possibility for process enhancement is reflected in repairing a process model, i.e., paths from a process model that are not discovered in a process execution log, can be removed to better reflect the information of a log [vdA16]. This is combined with the results of conformance checking and a process model. If the behavior of the process execution log is not matching the process model for example, the model can be adapted,

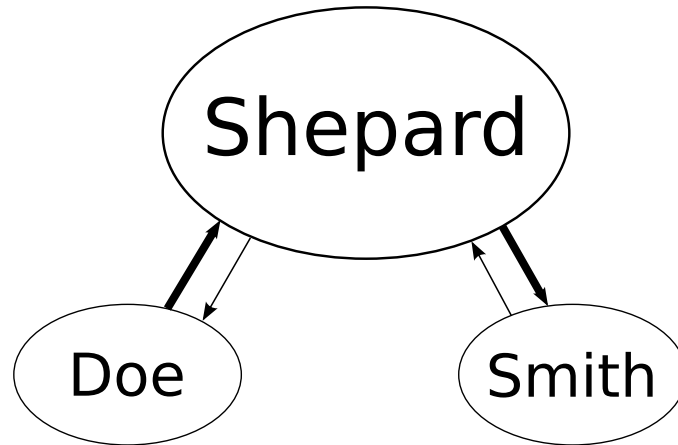


Figure 2.5: Example of a small social network for the bottom path of the process model depicted in Figure 2.1.

by adding and removing activities to the process model automatically. The goal is that a user-set threshold of traces is fitting the behavior of the process model afterwards. It is also possible to remove specific optional paths from a process model if the activities in this path are never executed according to a process execution log.

2.5 Concept Drift in Process Mining

A sudden shift, induced by, for example, a new legislation, can cause a change in a process. This is followed by a change in the process execution and the process execution logs. These changes are often not transferred to the process model, i.e., the new process instances are not fitting the already discovered process model based on the previous process instances. Therefore conformance checking algorithms in this case, are often detecting non fitting process instances. This phenomenon is called a concept drift [WK96]. In [BvdAŽP11], an approach to find concept drifts in process execution logs is discussed and the following features to interpret the relationships of events are introduced.

- **Relation Type Count.** Defines a vector for every event, containing the number of events that always, sometimes and never follow a specific event.
- **Relation Entropy.** The average rate at which a specific relation is being created.
- **Window Count.** The count is defined for a specific relation, like the follows relation, on a give length.
- **J-measure.** Originally proposed by Smyth and Goodman [GS91], to calculate the goodness of a rule, like b follows a. This is done with cross-entropy of two events and a specific windows size.

Concept drifts, are detected by splitting the log into smaller sub-logs and finding the point of change through statistical tests, like the Kolmogorov-Smirnov test and the Mann-Whitney U test.

Four different types of concept drifts are mentioned in [BvdAŽP11] and are explained in the following. An **incremental drift**, reflects only small changes in the new process model, i.e., a new additional event in the process model. Figure 2.6 presents a process model similar to Figure 2.1 but with an additional activity *FT*, reflecting a test for a flu infection, since an infected patient could be the source for severe problems for other patients in the same room.

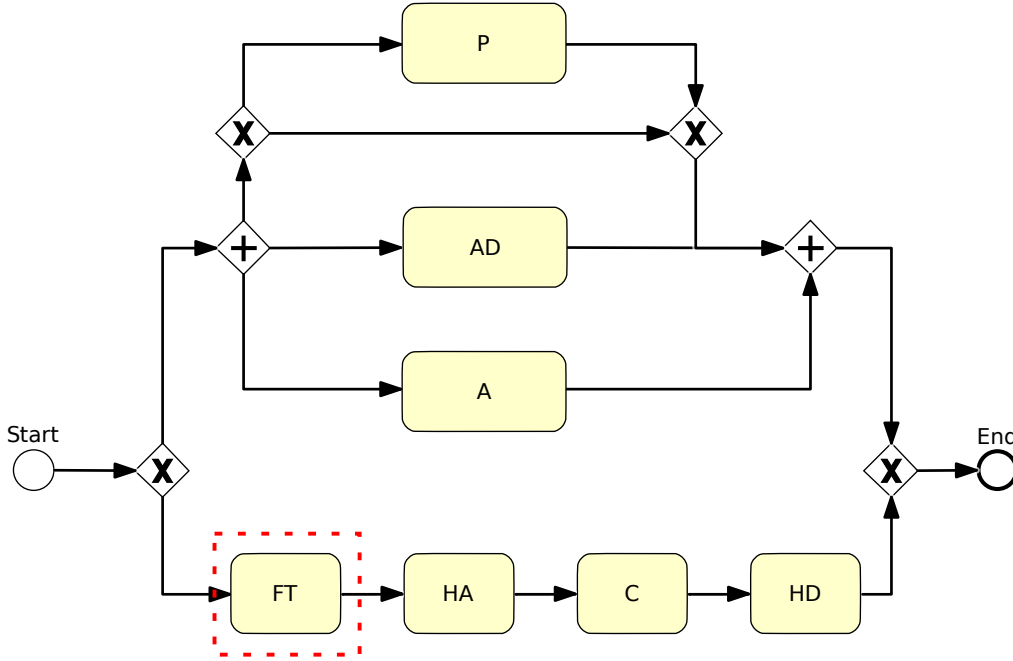


Figure 2.6: Example of an incremental drift to the process model in Figure 2.1. The red dashed line marks the additional event.

A **recurring drift**, reflects the use of a previously used process model for a time, before switching to another process model again. This is often due to seasonal effects, e.g., a different process model is used in the winter than in the summer. Figure 2.7 presents an example for a recurring drift. During the summer the process model from Figure 2.1 is used, while in the flu season, typically winter, the model from Figure 2.6 is used.

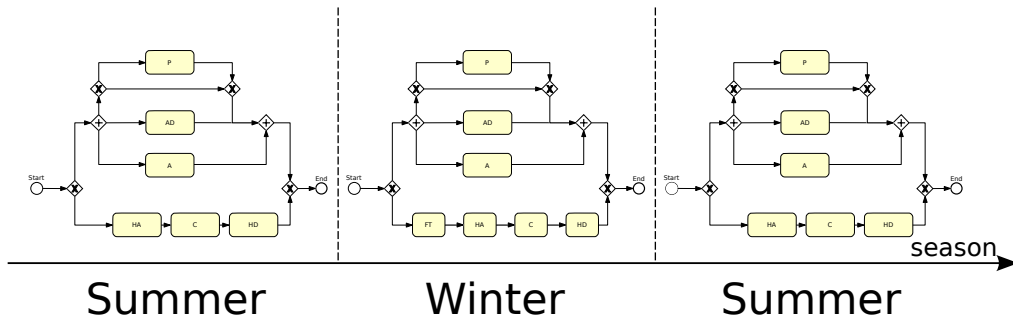


Figure 2.7: Example of a recurring drift. Two process models alternate depending on the season.

2 Related Work

A **gradual drift** occurs, when currently active process instances are still using the previous process model, but newly started process instances are using a new process model, i.e., patients admitted before the flu season do not have to be tested for the flu and follow the old process model, while new patients are tested for flu.

Contrary to this drift, is the **sudden drift**. When a sudden drift occurs, all currently active process instances are stopped, and have to be adapted to fit the behavior of the new process model, i.e., if the flu is already present in a medical care unit, all patients have to be tested for flu to contain the infection.

As already mentioned, these concept drifts can be detected, but the reason for the concept drift is unknown. The approach from [AvZQ⁺21] aims to mitigate this, by analyzing the different perspectives of a process execution log, e.g., the workflow and the resource perspective for example. After a concept drift has been detected, two perspectives are transformed to time series and are checked for drifts as well. If drifts are detected, the cause-effect is analyzed to see if a drift in one perspective is the cause for the drift in the other perspective.

Prediction in process mining is sometimes accomplished by using machine learning algorithms, which train a model based on a process execution log of a process [VdASS11]. Concept drifts can make these trained models inaccurate. The approach in [BRK20] aims to deal with concept drifts, by taken concept drift detection into account for the machine learning algorithms. The approach shows, that a retraining of the model as soon as the drift is detected, increases the accuracy, increasing the importance of online process mining algorithms.

The algorithms in this thesis build upon the related work that has been presented in this chapter. While there are already concept drift detection algorithms available, a clear definition of the drift types is still missing. Concept drifts are also not only occurring in the workflow perspective of a process, but can also occur in the data perspective of a process. This thesis aims at closing this gap and provide a collection of all documented and classified concept drifts and process models of a process. Conformance checking algorithms are usually only concerned with the control flow perspective as well and use a rather trivial cost function to calculate the alignments. This thesis emphasizes on this and provides new methods to annotate a process model for a more detailed cost function, as well as taking the data elements into account for the alignments. Last but not least, the cause for concept drift is only calculated using the data available in the process execution log in known approaches, but the cause can easily be outside of the process data, therefore a new method is presented to find the reason for a concept drift using data streams outside the process data.

3 Creating Data Sets for Process Mining

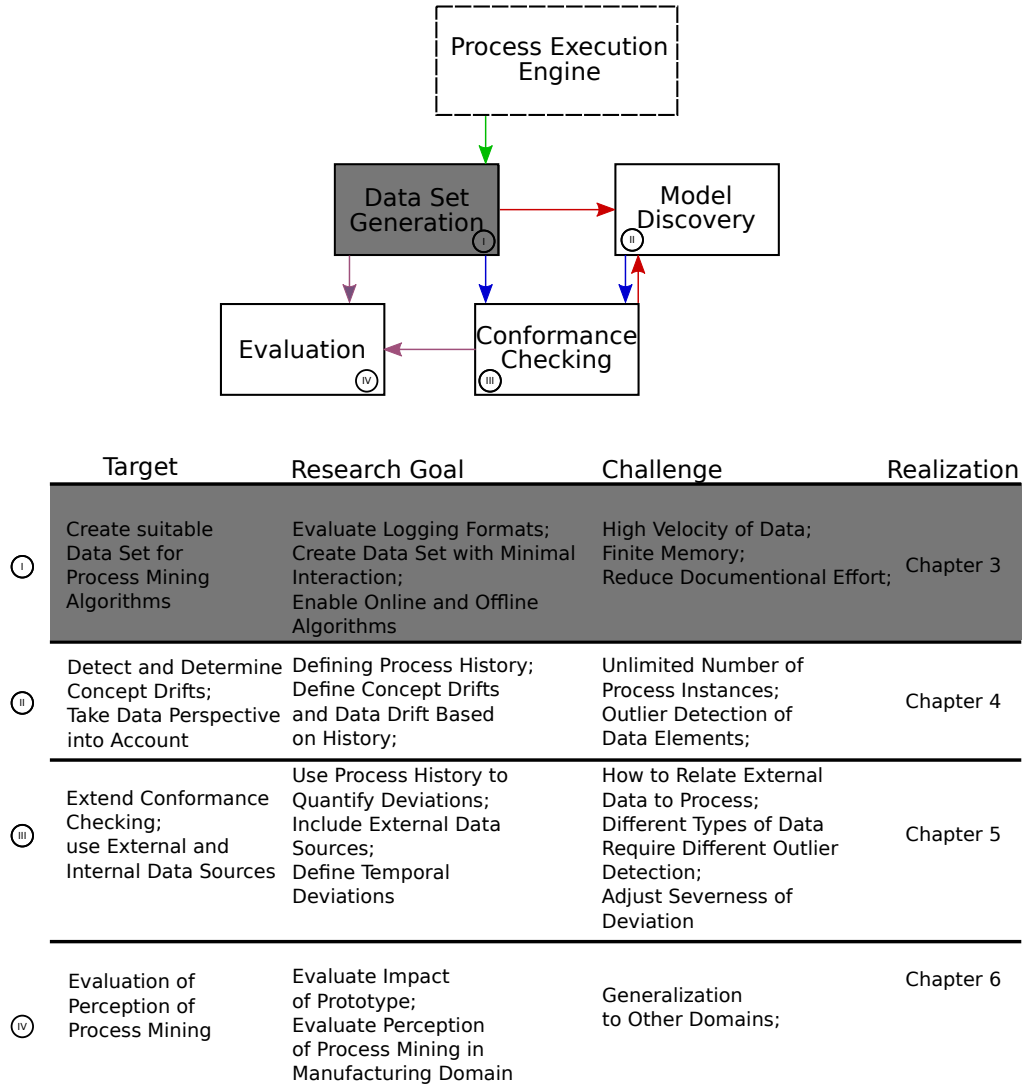


Figure 3.1: Overview of Targets, Research Goals, Problems of this thesis. The features for this chapter are marked.

This chapter addresses objective ① in Fig. 3.1, i.e., the generation of suitable data sets to apply process mining algorithms on, e.g., data sets that can be used for process mining algorithms either after or during the execution of a process without any transformation algorithms needed.

Process mining algorithms provide useful insights into the execution of processes. In order to apply process mining, a suitable data set is required. This chapter focuses on the

generation of such a data set and is split into three sections. . As mentioned in Chapter 1, the following three research questions have been derived for the generation of data sets.

- RQ (1a) How should a data format for process mining algorithms be designed to be created during the execution?
- RQ (1b) How to generate data sets directly at the execution of a process for online process mining algorithms?
- RQ (1c) How to improve the quality of the data set and reduce the error-proneness?

Section 3.1 tackles RQ (1a) and focuses on the properties of a data set for process mining, which is able to be generated during the execution of a process. As described in the previous chapter, there are two different input sources for process mining algorithms available, process execution logs and event streams. Listing 3.1 provides an overview of an example process execution log in the XES XML format [VBDA11] for one patient with severe symptoms following the process depicted in Figure 1.1. The log is starting with a log node, followed by a trace. The name of the trace reflects the name of a process instance, e.g., the name of a patient and provides important data elements, like the severity of the symptoms. A trace contains a number of events, describing the executed activities in a process instance. In this example, the start point and the end point of an activity is logged and distinguished by the lifecycle attribute of an event. XES XML is the most common adaption of the XES standard and offers a structured log which is human readable. An event stream, typically used for online process mining algorithms, consists of events similar to events in a process execution log. The events contain information of the related process instance, i.e., a trace, of the process. Contrary to a process execution log, the complete information of a process is not known since a stream is infinite, i.e., an endless number of events need to be processed. This imposes several challenges, i.e., a high velocity of events in an event stream, requires the algorithms to process one event in a short amount of time and an unlimited number of events cannot be stored, since memory is finite. The event stream is also following the XES standard [VBDA11]. Listing 3.2 provides an excerpt of an event stream, following the process depicted in Figure 1.1. As can be seen, the event stream does only contain events referring to the start of the “Cholecystectomy”, because apparently the activity is still being executed and no completing event has been detected in the event stream. Note that the trace information in the event stream is stored within the events as `concept:instance`¹. Section 3.1 focuses on the feasibility of a suitable data set for process mining algorithms. It focuses on providing a light weight human readable format with a consistent performance in logging process instances, which can then be directly used for process mining algorithms, online and offline.

```

1 <log xmlns="http://www.xes-standard.org/" xes:version="2.0" xes:features="nested-attributes">
2   <trace>
3     <string key="concept:name" value="John Doe" />
4     <int key="age" value="31" />
5     <string key="severity" value="severe" />
6     <event>
7       <string key="concept:name" value="Hospital Admission" />
8       <string key="lifecycle:transition" value="start" />
9       <string key="org:resource" value="Jane" />
10      <date key="time:timestamp" value="2021-09-10T13:00:12" />
11    </event>
12    <event>
13      <string key="concept:name" value="Hospital Admission" />

```

¹<https://www.tf-pm.org/resources/xes-standard/about-xes/standard-extensions/concept>

```

14 <string key="lifecycle:transition" value="complete" />
15 <string key="org:resource" value="Jane" />
16 <date key="time:timestamp" value="2021-09-10T13:28:56" />
17 </event>
18 <event>
19 <string key="concept:name" value="Cholecystectomy" />
20 <string key="lifecycle:transition" value="start" />
21 <string key="org:resource" value="Joan MD" />
22 <date key="time:timestamp" value="2021-09-12T09:22:51" />
23 </event>
24 <event>
25 <string key="concept:name" value="Cholecystectomy" />
26 <string key="lifecycle:transition" value="complete" />
27 <string key="org:resource" value="Joan MD" />
28 <date key="time:timestamp" value="2021-09-12T10:29:00" />
29 </event>
30 <event>
31 <string key="concept:name" value="Hospital Dismissal" />
32 <string key="lifecycle:transition" value="start" />
33 <string key="org:resource" value="Jane" />
34 <date key="time:timestamp" value="2021-09-16T15:28:56" />
35 </event>
36 <event>
37 <string key="concept:name" value="Hospital Dismissal" />
38 <string key="lifecycle:transition" value="complete" />
39 <string key="org:resource" value="Jane" />
40 <date key="time:timestamp" value="2021-09-16T15:52:56" />
41 </event>
42 </trace>
43 </log>

```

Listing 3.1: Example XES log file snippet for one process instance with severe symptoms for process depicted in Figure 1.1

```

1 <string key="concept:name" value="John Doe" />
2 <int key="age" value="31" />
3 <string key="severity" value="severe" />
4 <event>
5 <string key="concept:instance" value="John Doe" />
6 <string key="concept:name" value="Hospital Admission" />
7 <string key="lifecycle:transition" value="start" />
8 <string key="org:resource" value="Jane" />
9 <date key="time:timestamp" value="2021-09-11T13:00:12" />
10 </event>
11 <event>
12 <string key="concept:instance" value="John Doe" />
13 <string key="concept:name" value="Hospital Admission" />
14 <string key="lifecycle:transition" value="complete" />
15 <string key="org:resource" value="Jane" />
16 <date key="time:timestamp" value="2021-09-11T13:28:56" />
17 </event>
18 <event>
19 <string key="concept:instance" value="John Doe" />
20 <string key="concept:name" value="Cholecystectomy" />
21 <string key="lifecycle:transition" value="start" />
22 <string key="org:resource" value="Joan MD" />
23 <date key="time:timestamp" value="2021-09-13T09:22:51" />
24 </event>

```

Listing 3.2: Example XES Event Stream for one process instance with severe symptoms for process depicted in Figure 1.1

Section 3.2 focuses on RQ (1b) and establishes a system that allows process instances orchestrated by a process execution engine, to directly generate data sets for process mining while a process instance is being executed. Every time an activity is executed, a XES event is generated which can be used immediately. Using this system, process mining algorithms can be applied directly on an event stream of a process or on process execution logs, after the process instances are completed.

Section 3.3 focuses on RQ (1c) and introduces a novel approach to integrate human resources in the generation of a data set suitable for process mining. A process execution engine as discussed in the previous Section 3.2, allows the generation of a data set suitable for process mining, but it is still not clear how human resources can interact with this system, i.e., how can human resources execute a task in a process instance without increasing their effort with tedious tasks. Currently, human resources tend to fulfill their

tasks, but create the documentation of these tasks later, e.g., at the end of their shift. This leads to incorrect timestamps in the data set or even missing information. The approach presented in this section, enables human resources to directly interact with the process execution system using NFC tags, thus documenting every task when it is being executed leading to a data set with higher quality, e.g, correct timestamps. An additional benefit of this system is the reduction of documentation effort by human resources, since the data set generated by the process execution engine can be transformed to a standard documentation sheet. This can lead to a high acceptance of human resources for introducing a system like this.

A selection of text, figures and tables within this chapter is based on the following publications.

Stertz, F., Rinderle-Ma S., Hildebrandt T., Mangler J.: Testing Processes with Service Invocation: Advanced Logging in CPEE. In: 14th ICSOC Workshops (2016), Pages: 189-193
https://doi.org/10.1007/978-3-319-68136-8_22

Stertz, F., Mangler J., Rinderle-Ma S.: Balancing Patient Care and Paperwork Automatic Task Enactment and Comprehensive Documentation in Treatment Processes. In: Enterprise Model. Inf. Syst. Archit. Int. J. Concept. Model. (EMISA) Vol. 15, 2020, Pages: 34-48

3.1 Data Set properties and format

A plethora of process mining algorithms exist currently. There are offline process mining algorithms, i.e., after the execution of a process on a process execution log [vdA16]. There are also online process mining algorithms [MBCS13, vZBH⁺17, BC17]. Online process mining is applied to an event stream instead of a process execution log. Both of these input formats typically rely on the Extensible Event Stream (XES) [IEE16] format. A detailed description of both input formats is presented in Chapter 2.

The ex post setting is usually not time-critical, since the execution of the process is already finished. By contrast, the application of process mining techniques during run time can be quite time-critical, in particular, in high-volume settings with lots of running process instances producing a vast amount of events at the same time, thus an event should be already processed before the next event can be handled in the event stream. These online settings are highly relevant in order to monitor the compliance of process instances [LMM⁺15]. Thus computational time has to be short in order to react as soon as possible to emerging irregularities.

The **XES format** is commonly represented as an **XML** file, which is easily human-readable and can be processed in common open-source tools like ProM. But this aspect of XES constitutes a **potential bottleneck** at the same time.

Log files typically have two critical access patterns, where time is of the essence. The first access pattern is when an information system or workflow engine writes or updates the corresponding log file. The second access pattern is when process mining algorithms

are to be applied and therefore a log file has to be loaded into memory. In the second case, the memory restricts the amount of data that can be analyzed.

Writing log files directly when an event is occurring, does not seem to be the case in the current situation. XES files seem to be generated after process execution from more efficient internal representations, like databases [dMvdAR15].

Most existing widely-used log formats (NCSA Common Log Format², Extended Log File Format³), try to store a log message as fast as possible to not disrupt the actual work of the server. Further goals are to keep the log messages as compact as possible, to produce minimal system overhead during writing the message, and to provide decent extensibility regarding custom information. Thus these log formats typically are specified so that entries can be appended, which has the advantage that no file locking is necessary.

Exactly these two properties, the appending of entries so the file does not have to be locked and creating compact messages with minimal overhead concerning time efficiency can be improved. Currently, by using XML representations it is necessary to parse the entire file for adding entries to XML files and a write lock has to be set, thus leading to the question, is there another file format already available in which the XES standard can be represented without losing the advantages of XML?

In the following, an approach is proposed to keep the XES meta-model specification as is, but use a different representation, i.e., YAML. YAML is as human readable as XML [SH12, CGG⁺07] and can be used by many programming languages. Log messages can just be appended at the end of a file without needing to parse the entire document. Because it is not necessary to parse the whole tree structure like in XML, to verify its consistency and syntactic correctness, lots of memory can be saved.

This section in the following describes the structure of the YAML representation using the XES format and a prototypical implementation.

In Section 3.1.1, the XES-YAML approach is introduced together with its features. In Section 3.1.2, the approach is evaluated and the conclusion and a discussion is in Section 3.1.3.

3.1.1 Designing the XES-YAML Approach

To improve previously described disadvantages of logging in XML, other popular formats are investigated in the following, i.e., the aforementioned NCSA log format family, JSON [Cro06], and YAML. For the selection process of a logging format, it is important that any other chosen format than XML, still needs to be able to implement the XES standard, thus a conversion between both formats should be possible.

For the scenario of process logging, the **NCSA log format** has one severe disadvantage: as every entry, in this case most likely an event, is stored in one line, the format becomes hard to read for humans, and special escaping is required to store all the information required. Although it can be extended, it seems not a suitable candidate.

JSON [Cro06] is very suitable for representing the deeply structured information that the XES meta-model provides, but has the same disadvantages as XML: the whole file has to be read into memory to ensure syntactic consistency and thus access the data. Furthermore comments are not possible in the JSON format, and schema languages for validation are obscure.

²https://en.wikipedia.org/wiki/Common_Log_Format

³<https://www.w3.org/TR/WD-logfile.html>

Table 3.1: Comparison of different log formats for XES serialization. **Well structured** reflects if the data format can easily be checked for a specific format. **Comments** reflects the ability to make comments in the log file. **Append without Parsing** describes if the format is able to be extended without parsing the complete file format.

	XML	YAML	NCSA	JSON
Well structured	yes	yes	no	yes
Comments	yes	yes	no	no
Append without Parsing	no	yes	yes	no

YAML [BKEI05], shares the advantage of being structured and readable with JSON and XML, but has none of the disadvantages: as the format is not structured through *begin* and *end* tags, but rather through indentation and special characters, its consistency (and also conformance to schemas) can be checked on-the-fly for parts of documents.

Thus YAML is selected as the alternative format to represent the XES meta-model. The table 3.1 shows a comparison of the different log formats and their In the following, the corresponding XES-YAML representation of the core XES-XML concepts are introduced.

XML is typically structured like a tree. The root node being the log node

```
1 <log xmlns="http://www.xes-standard.org/" xes:version="2.0" xes:features="
  nested-attributes">
```

in XML equals

```
1 log:
2   _xmlns="http://www.xes-standard.org/2
3   _xes.version="2.0"
4   _xes.features="nested-attributes"
```

The standard extensions are represented like this:

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

This turns out in YAML as:

```
1 extension:
2   -
3     _name: Time
4     _prefix: time
5     _uri: 'http://www.xes-standard.org/time.xesext'
6   -
7     _name: Concept
8     _prefix: concept
9     _uri: 'http://www.xes-standard.org/concept.xesext'
```

The classifier would work with the same concept.

A trace containing a name is represented in XML as follows:

```
1 <trace>
2   <string key="concept:name" value="John Doe"/>
3 </trace>
```

In YAML, a trace can be expressed as:

```

1 trace:
2   string:
3     _key: 'concept:name'
4     _value: 'John Doe'

```

Events follow the same principle. An event is represented in XML as follows:

```

1 <event>
2   <string key="concept:name" value="Hospital Admission"/>
3   <string key="lifecycle:transition" value="start"/>
4   <string key="org:resource" value="Jane"/>
5   <date key="time:timestamp" value="2021-09-10T13:00:12"/>
6 </event>

```

The corresponding representation in YAML looks like:

```

1 event:
2   'concept:name': 'Hospital Admission'
3   'lifecycle:transition': 'start'
4   'org:resource': 'Jane'
5   'time:timestamp': '2021-09-10T13:00:12'

```

The above examples show that translation of XES-XML concepts to XES-YAML is simple and straightforward.

One advantage of YAML, is the performance during the creation of execution logs. Basically – like XML – YAML [BKEI05] can be easily parsed in most programming languages. For the following example, every process instance creates its own log file. To create a process execution log with all instances, these log files need to be merged, where each log file is reflected as a trace for the whole process.

In principle, appending data to XML files is not straightforward, due to the end-tags. Appending, for example, an event to a trace in XES-XML is not as simple as appending an event to the end of the file. Instead it can be compared to inserting characters before the end of the file, specifically, before the `</trace>` and `</log>` tags, which in turn are allowed to have different indentations, or can even occur in the same line. Thus the file has to be read completely, in order to find out the correct point of insertion. The insertion then requires file locking at file-system level, which is rather inefficient if many such operations occur.

In YAML you can just append an event at the end of the file without reading or parsing the document. File-system level append operations are VERY efficient and require no file locking. An example of a YAML log file can be seen in Lst. 3.3.

Every `---` marks the start of a new part in a YAML file. That is because every time a new event arrives at the logger, a new YAML file gets appended to the old one. This file can then be loaded again through a simple script, see for example Lst. 3.4.

```

1 ---
2 log:
3   extension:
4     time: http://www.xes-standard.org/time.xesext
5     concept: http://www.xes-standard.org/concept.xesext
6     organisational: http://www.xes-standard.org/org.xesext
7     lifecylce: http://www.xes-standard.org/lifecycle.xesext
8   global:
9     trace:
10       'concept:name': __INVALID__
11     event:

```

```

12     'concept:name': __INVALID__
13     'lifecycle:transition': complete
14     'time:timestamp': ''
15   trace:
16     'concept:name': 'John Doe'
17     'trace:id': '674'
18   ---
19   event:
20     'trace':id: '674'
21     'concept:name': 'Hospital Admission'
22     'lifecycle:transition': 'start'
23     'org:resource': 'Jane'
24     'time:timestamp': '2021-09-10T13:00:12'
25   ---
26   event:
27     'trace:id': '674'
28     'concept:name': 'Hospital Admission'
29     'lifecycle:transition': 'complete'
30     'org:resource': 'Jane'
31     'time:timestamp': '2021-09-10T13:28:56'

```

Listing 3.3: Example of a YAML XES File

Furthermore, in order to show how to efficiently read a YAML file the following pseudo code is provided:

```

1 def load_yaml_from_buffer(buf)
2   unless buf.empty?
3     x = YAML.load(buf)
4   end
5   buf.clear
6   return x
7 end
8
9 def loop_xes_file(fname)
10  buf = ""
11  file = File.open fname
12  file.each do |line|
13    if line == "---\n" || file.eof?
14      full_event = pbuf(buf)
15      ### work on single events
16    end
17    buf += line
18  end
19 end
20
21 loop_xes_file "log.xes"

```

Listing 3.4: Pseudocode for loading a XES-YAML file Into Memory

As can be seen in Lst. 3.4, two simple functions can be written to read the XES file. The function *load_yaml_from_buffer* is merely a helper that loads a YAML object out of a string buffer. The **loop_xes_file** function loops the file line by line (line 12 in the source) and triggers the YAML parsing whenever a new part (see explanation above) occurs. Line 15 in the source shows where the code to work on individual events can be inserted. Line 21 shows how **loop_xes_file** is called with a XES file.

The code in Lst. 3.4, even when parsing a million events, never uses more memory than the maximum size of a single event.

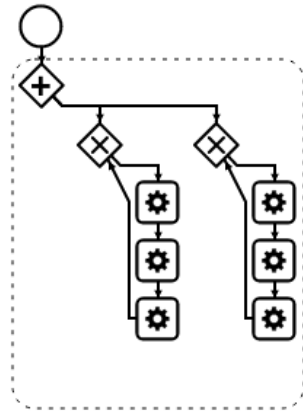


Figure 3.2: Example Process containing 2 parallel branches with 3 activities each in a loop.

3.1.2 Evaluation of XES-YAML File Format

For the evaluation of the XES-YAML file format, a simple process is created, which basically runs forever. The goal for this evaluation, is to test the feasibility and the performance of this file format. This process can be seen in Fig. 3.2. The test machine had 2 CPUs, each with 2.2 GHz and 8 GB RAM. The OS was Fedora 25 Server Edition. The start point for measuring the time spent for saving one event was when the log file was opened. The end point for measuring the time spent for saving one event was when the log file was closed again. A first log file was created in the XES-XML format that contains 1000 events. A second log file was created in the XES-YAML format that contains 1000 events as well. The time spent for saving one event was extracted and saved and evaluated with R [T⁺13]. The time spent for each event was then plotted against the number of the activities already stored in the log file.

As can be seen in Fig. 3.3, while the time used for saving an event with the XML representation is growing linearly, the time used for saving an event with the YAML representation remains constant.

Table 3.2 shows, that as expected, the time spent for storing an event is lower using the YAML representation. The time difference between XML and YAML does not seem much, but since a workflow engine typically has multiple process instances running in parallel, this time difference can make a big difference, since each process instance creates a separate log file, which can be merged after the execution for process mining algorithms. For this evaluation, only one process instance is executed at a time, so that the results can be compared at hand. For 1000 events, the XML representation also consumes more space on the hard disk with 547 *kibibytes*⁴ compared to 376 kilobytes.

XES-XML's disadvantages concerning runtime of process mining techniques or the creation of log files usually are not problematic for offline process mining. However for online process mining, time is a crucial resource. XES-YAML can reduce the problems of XES-XML, still following the meta model of XES. Moreover, XES-YAML event files can be created in a linear time during execution.

⁴SI unit (Système international d'unités); 1 kibibyte is 1024 bytes, cmp. 1 kilobyte is 1000 bytes.

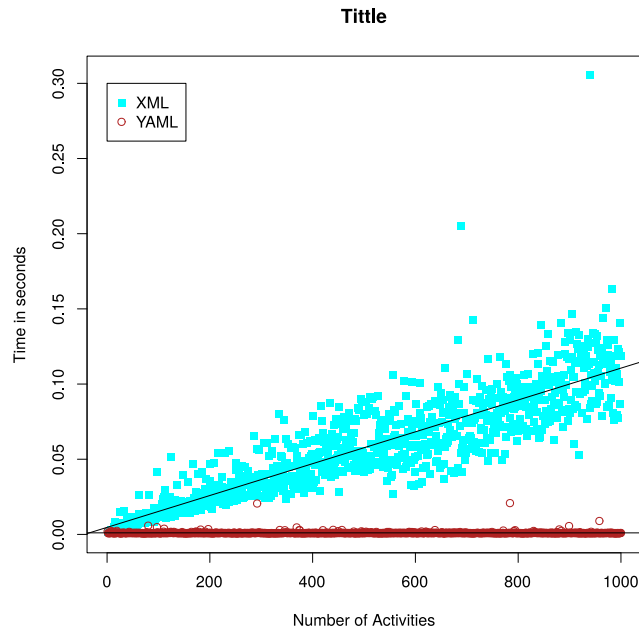


Figure 3.3: Time consumption for storing events in XES-XML and XES-YAML

Representation	XML	YAML
Minimal time in seconds	0.0014	0.0005
Maximal time in seconds	0.3058	0.0208
Average time in seconds	0.0576	0.0010
Filesize in kibibytes	547	376

Table 3.2: Performance Cornerstones

3.1.3 Results of the XES-YAML Approach

A new XES format has been elaborated to improve on its current XML representation. The disadvantages of XML are: (1) performance problems when directly writing big logs from process engines, and (2) memory problems when reading big logs. To pave the way for modern, future real-time process mining techniques, a faster approach for representing and loading XES is needed.

The approach presented, does not interfere with the XES meta-model, but instead relies on replacing the XES-XML representation with a XES-YAML representation.

The advantages of YAML over other well known formats, like JSON and NCSA, are discussed and the XES-YAML approach is compared with other related work concerning the creation and processing of log files. Although the XES-YAML approach is much simpler and straightforward than other approaches, it solves the problems at hand in a more efficient and elegant way.

3.2 Creating Process Mining Data Sets during the Execution of a Process

Simulating and testing business processes is crucial for different reasons such as improving process performance or detecting problems with web services. Process simulations are performed based on stochastic information and without considering the behavior of invoked services. As a consequence, the obtained results do not reflect the behavior that can be observed later during process execution.

Instead of simulating process executions, the system described in this section, creates and runs process instances in a testing environment based on the Cloud Process Execution Engine (CPEE)⁵. The CPEE – a lightweight, modular, and service oriented process engine - has proven its maturity in many application scenarios [MRM14]. Working with a process engine enables the testing of process instances together with the invoked services by calling real services, which may either be the same services used in productive execution, or demo services simulating a specific behavior for testing purposes. On top of more realistic testing results, the tested process instances can be directly put into “production mode” without export / import between execution engine and simulation environment, necessitating possibly quite some additional reworking of the simulated processes before being able to execute them. This section focuses on the suitability of a process execution engine to generate data sets for process mining algorithms. In the following, the necessary steps and different options for testing process instances in the CPEE are described, starting with the import of a process model provided in BPMN notation and creating and assigning services that are to be executed. It is then shown how to spawn and execute process instances and how the generated process logs can be configured and retrieved. Application scenarios are discussed as a conclusion.

3.2.1 Process model creation

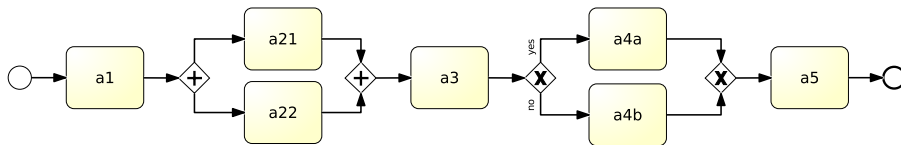


Figure 3.4: Process model containing one decision and one parallel gateway (modeled in BPMN, using Signavio)

At first, the process model will be initiated and executed has to be created. In the CPEE, a process model can be created either directly using a graph model or by importing BPMN process models created in, for example, Signavio⁶.

Figure 3.4 shows a small BPMN use case that can be imported. This model shows a parallel activity, at **a21** and **a22**, and a decision at **a4a** and **a4b**. This use case will be employed throughout this section.

Another option would be to compose the diagram directly in the CPEE Cockpit⁷. The result of both options can be seen in Figure 3.5. Each of these activities invokes a specific

⁵<http://cpee.org>

⁶<http://academic.signavio.com/>

⁷A simple frontend for the CPEE, that allows for creating and inspecting process images

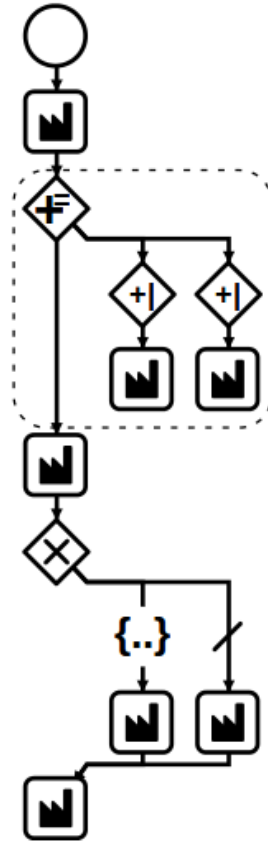


Figure 3.5: CPEE Process model

web service. The CPEE can orchestrate services that are reachable over the HTTP(S) or XMPP protocols, other protocols are easy to implement. When a service is finished, the next activity is marked active.

The goal is to create a log file for every process instance in the CPEE to simulate a huge amount of data in a short time. These logs are created using the XES format, YAML and XML, but any other format can be easily implemented.

3.2.2 Implementation

Every time an activity is enacted, manipulated, or finished, the custom logging extension is called to process this information. There are different types of events that can be logged.

- Activity/Called. Every time a service is invoked, usually after the last activity has been finished.
- Activity/Manipulated. When a service returns some data, this data is incorporated into the process context (assigned to variables, made available as potential input for future service invocations) which triggers this notification.
- Activity/Finished. When the invocation as well as the manipulation of process context is finished, a notification is pushed.

Usually for mining a process model, finding the control flow is most important. Therefore it has to be logged when a service reports back to the CPEE and an activity is finished. The main part of such log files are timestamps for every event. There are at least two possible options for timestamps.

Every time the process management system receives the information about an activity to be finished, the timestamp can be either (1) a timestamp provided by the service, which marks the exact time when work was finished, or (2) a timestamp generated by the process engine that includes the latency and time that is required for returning the data from a service.

Both options have their advantages when analyzing the result. Especially the first option (which is not very common) can be useful when huge quantities of data are transmitted over unreliable connections.

The developed logging service supports both options. If the services deliver a timestamp, the timestamp of the service will be used, but if no timestamp is delivered, the CPEE will create a timestamp and use this one for the log file. The minimum documentation of events are timestamps and a label of an event, but usually there is more information associated to an event. Another interesting point in logging information of the execution of a process instance are data values. For each service there is input and output. Currently, there is no standardized way of storing this data in the XES⁸ format, but it allows to be extended in a flexible way. Thus two lists per event are created in the log file to store this additional information (see Listing 3.5).

```

1 if @log_hash.has_key?(:data_received)
2   @log_hash[:data_received].delete_if do |e|
3     if e.keys[0]=="timestamp"
4       event.add 'date', :key => "time:timestamp", :value => e.values[0]
5       time_added=true
6     else
7       false
8     end
9   end
10  if @log_hash[:data_received].length > 0
11    list = event.add 'list', :key => "data_received"
12    @log_hash[:data_received].each do
13      |e| list.add 'string', :key => e.keys[0] , :value => e.values[0]
14    end
15  end
16 end

```

Listing 3.5: Example of storing received data

This allows for more than mining the process model, like decision point mining [RvdA06] for specific events. After the instance is executed and the status is finished, the log file can be retrieved directly from the CPEE.

```

1 <event>
2   <string key="concept:name" value="a22"/>
3   <date key="time:timestamp" value="2016-07-14T12:56:36+02:00"/>
4 </event>
5 <event>
6   <string key="concept:name" value="a3"/>
7   <date key="time:timestamp" value="2016-07-15T12:56:36+02:00"/>

```

⁸<http://www.xes-standard.org>

```
8   <list key="data_received">
9     <string key="decision" value="yes"/>
10  </list>
11 </event>
12 <event>
13   <string key="concept:name" value="a4a"/>
14   <date key="time:timestamp" value="2016-07-16T12:56:36+02:00"/>
15   <list key="data_send">
16     <string key="value" value="yes"/>
17   </list>
18 </event>
```

Listing 3.6: Example of Log File

Listing 3.6 shows an example of a log file for the process model described Figure 3.4. For each event the name and the timestamp is stored and for some events the used data is stored as well if available.

3.2.3 Application Scenarios

The logs retrieved from the CPEE can be utilized for process mining algorithms, on- and offline, i.e., by listening to the notifications stream of events from the CPEE directly or by reading the created log file per process instance. The creation of other log types is possible, for example, change logs that are crucial for recommending changes [KMS⁺15b] or change propagation logs that enable the prediction of change propagation behavior in distributed process settings [FRMI14]. The advantages of the CPEE combined with the new logging services are in creating a fast test environment for services, which can be deployed into *real world* production environments with minimum effort.

Section 3.1 and Section 3.2 described a suitable format for process mining data sets and how they can be generated. It is important to note, that the logging approach in this form automatically logs activities that are performed by services. However, in various domains, activities are performed by human resources without any computer interaction at all. In the next section, a novel system is established, that allows human resources to interact with a process execution without increasing the complexity of their work by adding additional tasks to log their activities.

3.3 Creating Accurate Data Sets for Process Mining with Human Resources Involved

For creating accurate data sets during the execution of process instances with human actors involved, a closer look is taken into the care domain. Processes in the care domain, are typically executed by human resources, i.e., nurses in a nursing home, without any computer interaction, however the log of a process instance equals a good documentation of the tasks the staff in a nursing home is executing. Thus the care domain is an ideal subject for creating process mining data sets with human resources involved.

Basically, documentation in the care domain is carried out based on paper-based or electronic health records where “*each have their drawbacks in the real practice of nursing documentation*” [AZAMBH18]. On the one side, paper-based documentation is found to “*not meet the requirements of high-quality documentation and communication among*

healthcare providers, because it is time-consuming, repetitive and inaccurate” [AZAMBH18]. On the other side, the application of electronic health records is still limited and its impact on quality of documentation has not been fully proven yet [AZAMBH18]. As stated in [LR07], (health) care in general craves for process-aware solutions. The process awareness leads to a task-oriented instead of data-centered view on the residents. This, in turn, facilitates the continuity of the documentation at the granularity of a care task and is hence promising for decreasing documentation effort and increasing documentation quality. This approach is taking a process-aware care solution and the results from the previous sections as a starting point with the main goal to automatically log the tasks executed by the staff of a nursing home, while not increasing the documentational effort of the staff members.

3.3.1 Care scenario and contributions

As an example for this approach, the morning routine in a nursing home⁹ is depicted in Fig. 3.6. The control flow is modeled using Business Process Modeling and Notation (BPMN).

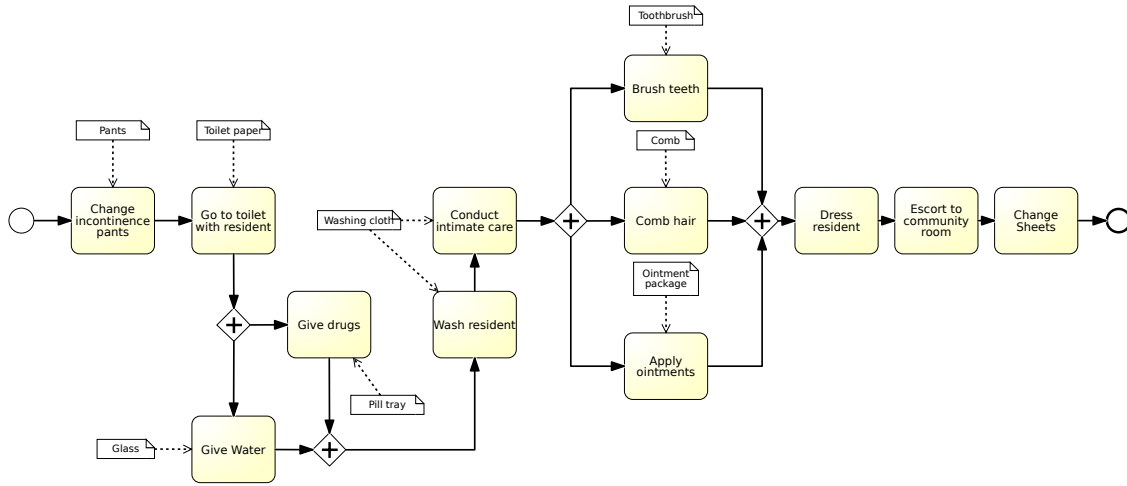


Figure 3.6: Daily morning care routine (BPMN notation using Signavio)

For each of these tasks, typically, the nurse has to document that i) the task has been completed and ii) data connected to the task (e.g., the task was conducted by Nurse A at time T with drug dosage D). A process-aware solution where care processes are modeled, enacted, and executed through a PAIS, provides essential support for automation of i) and ii). More precisely, a PAIS offers work lists to the nurses. These work lists contain the care tasks that are due in their context (i.e., for a specific resident with his/her data necessary for the task). One example is task *Give drugs* where the dose for a specific resident is indicated in the work list. Hence, overall, PAIS support could already facilitates automatic and continuous care documentation.

However, this approach aims at going one step further, exploiting the fact that typically many tasks in a care processes utilize physical objects (denoted as *care utilities* in the

⁹The process is delineated by international nursing guidelines such as of the North American Nursing Diagnosis Association (NANDA) [BM98] and national laws.

following). Take the morning routine as an illustration¹⁰. Nine out of twelve care tasks involve care utilities, for example, *Pants* for task *Change incontinence pants*.

Care utilities are key to enable automatic completion of tasks, equipped with some set of data values. In detail, care utilities are equipped with NFC tags storing precise information about the care utility. Figure 3.7 shows a resident bed equipped with a NFC reader in an experimental setting at the lab of the Research Group Workflow Systems and Technology and a NFC tag on the bottom right corner on a small plastic lid. NFC tags can be applied on nearly any surface, are small, and are cheap to be produced. Every time a care utility equipped with such an NFC tag gets swiped in front of the NFC reading device, a message is sent.



Figure 3.7: Resident bed equipped with NFC reader and a NFC tag on a small plastic lid.

To realize this, tag readers are built into the nursing home residents' beds. Every tag reader is associated with the bed of one specific resident. Every time the reader detects a tag, it tries¹¹ to find a matching active task in the residents' care process, marks it as finished and documents the occurrence with a timestamp and the care utility. This can effectively reduce the time a nurse needs for the documentation of a resident and improve the quality of the documentation at the same time.

The requirements on the intended solution are elaborated based on selected use cases from the care domain. These use cases are derived from interviews with experts in the care domain. The artifacts, in turn, are elaborated based on the requirements and evaluated through a prototypical implementation (feasibility) and interviews with domain experts

¹⁰As BPMN does not support the modeling of physical devices, it was opted to model the care utilities as input data objects at the tasks where they are used.

¹¹If no task is found, the activity for the resident is logged with a note that it was either an emergency or an error.

(effectiveness). Note that this approach is a revised and extended version of [SMR17]. The extension includes a more comprehensive set of use cases, a more detailed and comprehensive solution design, and the automatic generation of paper-based documentation based on the logs.

The remainder of this section is structured as follows: Section 3.3.2 provides specific background information on NFC technology together with justification to employ NFC technology in the envisioned process-aware care solution. The conceptual solution based on nine care use cases is presented in Sect. 3.3.3. Section 3.3.4 describes the architecture of a process-aware care solution and presents its implementation and various design choices. The evaluation of the proposed solution is provided in Sect. 3.3.5. The initial solution has been designed and realized for the care domain. As discussed in Sect. 3.3.6, it can be transferred to other application domains such as manufacturing and logistics. In Section 3.4, a summary and outlook for future work is provided.

3.3.2 Background on NFC technology

In order to not increase the documentational effort for the nursing home staff, each care utility has to provide information about its existence and potentially its state. *Near Field Communication (NFC)* is one of the technologies for sending and receiving data wireless in close range. Alternative technologies would be Barcodes or RFID. In the following, pros and cons of the three technologies are discussed.

Barcodes: 1D barcodes store rather little information (13 digits) since usually additional information is then looked up in the database [KT05]. 2D barcodes, e.g., QR-Codes, can hold more information which can be decoded without accessing a database. Limitations of using barcodes are restricted usage (“line of sight” is needed), lack of reusability (barcodes are printed on the object), and that they cannot be edited.

Radio-frequency identification technology, RFID [Wan06] supports active and passive devices. Another advantage is that no “line of sight” is required. Moreover, RFID devices are cheap to produce, hence can be used at a large scale from an expenses point of view.

Near Field Communication, NFC [AB12] is used in many applications and devices such as Smartphones, Debit cards, and tickets for public transportation. NFC builds upon the RFID technology [AB12] and communicates on a very short range below 10cm. NFC can work with an active device, the NFC reader, and a passive object, like a tag or card. The reader can write and read information of the object. A big difference to RFID is the range. A RFID reader can create a larger electromagnetic field to automatically detect different objects going through it. Since the communication range of the NFC reader is quite low in comparison, there will not be any unwanted detection of different tags entering a room, which would create a flawed documentation. NFC tags are cheap to produce, very thin, and can be attached to any surface, like stickers.

In conclusion, NFC is chosen for the process-aware care solution presented in this work due to the following advantages:

1. *Ubiquitous availability*: NFC-capable devices can be found everywhere nowadays such as smart phones, toys, table computers, and even watches. This allows for an approach which is feasible to acquire and relatively easy to learn, since the devices are used in everyday lives.

2. *Better security:* Since the distance to communicate is only about a few centimeters, NFC is safer to use than RFID. The short range increases the level of difficulty to create unwanted detections and or malfunctioned detections.
3. *No unwanted detection:* Another advantage of the close communication range of the NFC technology is the elimination of unwanted tag detection. In a room of a resident are many care utilities. RFID creates a large field, where every tag will be detected. This is useful for warehouses, but in the nursing home, it is only necessary to detect the right NFC tag at the right time. The short communication range allows only one tag to be detected if it is right in front of the NFC reading device, thus allowing for a correct documentation.
4. *Usability:* Existing approaches state that “NFC technology was perceived as very intuitive and the information quality of each patient’s health status could be improved” for a project on documentation of the health status of patients [PML12]. Another approach utilizes NFC in nurse training, emphasizing the unobtrusiveness of the solution [FHBV11]. For not changing the daily routine of nurses too much, a way to unobtrusively register these tags with a PAIS during care activities is needed. Reading the tags is not intended to take longer than a few seconds and should happen right at the moment the care task is finished. Additionally, nurses are generally open to try out new technology as long as it helps reducing the documentary effort [DHK⁺10].

3.3.3 Conceptual solution design based on use cases

The solution design covers nine use cases that are relevant for automatic task completion and documentation in care processes.

Figure 3.8, gives a quick overview of the use cases, their actors and the relations between the use cases. Organizational tasks such as “Register resident” are usually conducted by the administrative staff of a nursing home. If the nursing home is small, care staff members can do organizational tasks as well. Use cases like “Write Utility Information” is a use case for more complex care utilities, which provide information on contradicting care utilities, for example two drugs that act as a blood thinner are contradicting each other, since the doubled amount of blood thinners will lead to complications. A detailed description of all use cases is following.

A use case description contains information about its *scope*, *level*, *primary actor*, *stakeholders and interests*, *preconditions*, *postconditions*, *realization* and *frequency of occurrence*. The *scope* of all relevant use cases is *system-wide*, i.e., the related information is relevant and visible not only for a certain task, but throughout the system. The *level* for all use cases is *user goal*, i.e., all use cases can be done completely by one staff member or nurse at one time, thus keeping the data consistent.

For each task that employs NFC-supported documentation, first of all, a corresponding NFC tag is registered in order to be used in the sequel (cf. Use case 3.3). Also, each resident has to be registered in the database (cf. Use case 3.4). Each resident is associated with one NFC reader attached to a bed, and each NFC reader is associated to at most one resident. This one-to-one relationship between residents and NFC tags is necessary for the assignment of events to the resident.

3.3 Creating Accurate Data Sets for Process Mining with Human Resources Involved

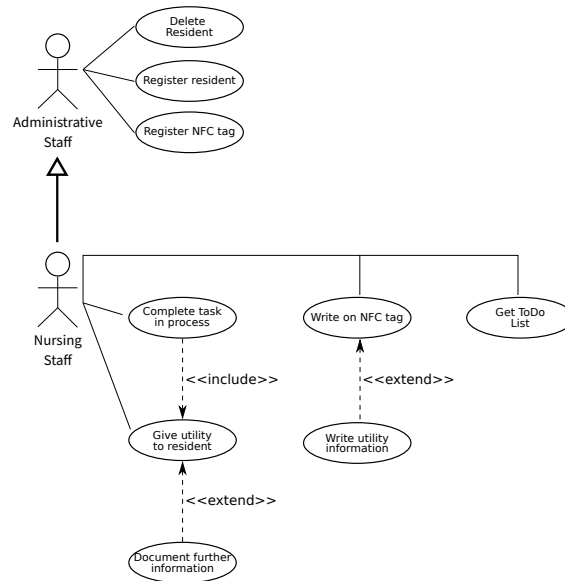


Figure 3.8: Use case Diagram showing the potential actors and use cases

Table 3.3: Use case: Register NFC tag

<i>primary actor</i>	staff of nursing home
<i>stakeholders and interests</i>	staff member (stakeholder 1) wants to register a NFC tag for documentation.
<i>preconditions</i>	ID of NFC tag not in database (see <i>realization</i>)
<i>postconditions</i>	unique ID for NFC tag in database (see <i>realization</i>)
<i>realization</i>	
<i>frequency</i>	often

It is also possible to write on a NFC tag (cf. Use case 3.5), specifically, the type of the connected care utility (e.g., a comb). The type can be *simple* or *complex*. For complex care utilities the NFC tag holds additional information, e.g., the dosage for utility drug. In the realization of the use case this distinction is made by invoking a subprocess.

If a care utility contains complex information this information is written onto the NFC tag as described in Use case 3.6. An example is painkiller Parkemed as care utility which can be administered in dosages 250mg and 500mg. Dosage is a critical parameter in the

Table 3.4: Use case: Register resident

<i>primary actor</i>	staff of nursing home
<i>stakeholders and interests</i>	staff member (stakeholder 1) wants to register a resident and relate him or her a NFC reading device.
<i>preconditions</i>	new resident has not been assigned a NFC reading device (see <i>realization</i>)
<i>postconditions</i>	NFC reading device is connected to the resident in the database (see <i>realization</i>)
<i>realization</i>	<pre> sequenceDiagram participant Staff as Staff (Nursing home) participant NFC as NFC reader participant System Staff->>Staff: Start Staff->>NFC: Hold Tablet pending resident ID in front of NFC Reading device NFC->>NFC: Read resident ID Note over NFC: Decision 1 NFC->>Staff: Show failure message Note over NFC: error reading tag NFC->>NFC: Read NFC reader ID Note over NFC: Decision 2 NFC->>Staff: Show failure message Note over NFC: error reading tag NFC->>System: Register resident System-->>NFC: NFC reading device or resident already connected </pre>
<i>frequency</i>	often

context of care utilities due to the prevention of misuse. Use case 3.6 can be generalized to other parameters. Nested information can be stored on a NFC tag as well, i.e., using containers. Containers also enable to store the information of more than one care utility on the NFC tag. The medical containers are usually prepared in the nursing room during quieter times like the night shift. To write the information on a NFC tag, a tablet in the nursing room is required. Tablet PCs can act as NFC reading and writing devices.

Often staff members want to retrieve the upcoming tasks, either for themselves or for a specific resident. Both can be initialized by reading the NFC tag assigned to the staff member or the resident respectively. As process technology is employed the upcoming tasks can be determined based on the to do list for the staff member or resident (cf. Use case 3.7). Knowing upcoming tasks, the staff member can optimize the personal work routine and hence avoid unnecessary tasks or waiting times during his or her shift. A requirement for the accessing the to do list is that the staff member is equipped with a tablet computer holding a NFC reading device, which displays the currently active tasks.

Use case 3.8 *Give Utility to Resident* realizes the automatic documentation of a care task. The care utility carries a NFC tag which is placed on the NFC reading device of the resident. Doing so the connection between resident (specifying the treatment process instances) and care utility (specifying the care task) is made. The NFC reader decodes the information read from the NFC tag. The system automatically documents that the care task has been completed for the resident with all necessary information such as timestamp and possibly dosage. Looking at the morning care shown in Fig. 3.6 together with care utilities, the automatic documentation would be conducted, for example, when the staff member hands the toothbrush to resident *Smith*. The system would then automatically

3.3 Creating Accurate Data Sets for Process Mining with Human Resources Involved

Table 3.5: Use case: Write on NFC tag

<i>primary actor</i>	care staff
<i>stakeholders and interests</i>	staff member (stakeholder 1) wants to connect a care utility to a specific NFC tag.
<i>preconditions</i>	NFC tag is not connected to a care utility in the database yet (see <i>realization</i>)
<i>postconditions</i>	ID of NFC tag is connected to a care utility in the database (see <i>realization</i>)
<i>realization</i>	<pre> sequenceDiagram participant System participant Staff participant NFC_reader as NFC reader Staff->>NFC_reader: Put Care Utility in front of NFC Reader NFC_reader->>NFC_reader: Read ID of care utility NFC_reader->>NFC_reader: Read ID of NFC tag NFC_reader-->>System: read error System->>System: Show failure message NFC_reader->>NFC_reader: Write ID of care utility on NFC tag NFC_reader-->>System: read error System->>System: Show failure message NFC_reader->>NFC_reader: Write ID of care utility on NFC tag NFC_reader-->>System: read error System->>System: Show failure message </pre>
<i>frequency</i>	often

Table 3.6: Use case: Write utility information

<i>primary actor</i>	care staff
<i>stakeholders and interests</i>	staff member (stakeholder 1) wants to connect a care utility to a specific NFC tag.
<i>preconditions</i>	utility type is complex; complex parameters (like dosage) need to be specified (see <i>realization</i>)
<i>postconditions</i>	dosage is written on NFC tag with ID of care utility (see <i>realization</i>)
<i>realization</i>	<p>The diagram illustrates the realization of the use case 'Write care utility ID on NFC tag' across three lifelines: Staff, System, and NFC reader. The process begins with the Staff actor initiating the 'Staff started writing on NFC tag' activity. This leads to the 'Write care utility ID on NFC tag' use case on the NFC reader lifeline. A decision diamond follows, branching into 'complex care utility' and 'simple care utility'. The 'complex care utility' path involves a 'Write complex information on NFC tag' use case on the NFC reader lifeline, which then triggers an 'NFC reader' activity on the Staff lifeline. This activity leads to a decision diamond for 'error writing care utility information'. If there is an error, the Staff actor shows a 'failure message' on the System lifeline. If successful, the Staff actor shows a 'failure message' on the System lifeline. The 'simple care utility' path also leads to a decision diamond for 'error writing care utility information'. If there is an error, the Staff actor shows a 'failure message' on the System lifeline. If successful, the process ends at a final state.</p>
<i>frequency</i>	often

document that task *Brush teeth* has been completed for process instance *Smith*.

Table 3.7: Use case: Get ToDo list

<i>primary actor</i>	care staff
<i>stakeholders and interests</i>	staff member (stakeholder 1) wants to avoid unnecessary tasks and waiting periods during shifts.
<i>preconditions</i>	—
<i>postconditions</i>	list of upcoming tasks is received (see <i>realization</i>)
<i>realization</i>	<pre> sequenceDiagram participant Staff participant System participant NFC_reader as NFC reader Staff->>System: Enter personal ID activate System System->>NFC_reader: Read ID from NFC tag activate NFC_reader NFC_reader->>System: ID stored on NFC tag deactivate NFC_reader System->>System: Retrieve ToDo List activate System System-->>System: no todo list found for ID deactivate System System->>System: Show failure message activate System System->>System: Display ToDo List deactivate System </pre>
<i>frequency</i>	often

Table 3.8: Use case: Give utility to resident

<i>primary actor</i>	care staff
<i>stakeholders and interests</i>	staff member (stakeholder 1) wants automatic documentation of care utility given to resident.
<i>preconditions</i>	resident is ready to receive care utility; NFC reading device for resident is registered; NFC tag of care utility is registered and has necessary information stored (e.g., dosage)
<i>postconditions</i>	resident successfully receives care utility; this is documented automatically in the system (see <i>realization</i>)
<i>realization</i>	<pre> sequenceDiagram participant Staff participant System participant NFC_reader as NFC reader Staff->>System: Put NFC tag of care utility on NFC reader of resident activate System System->>NFC_reader: Read NFC tag activate NFC_reader NFC_reader->>System: deactivate NFC_reader System->>System: Give resident care utility activate System System-->>System: read error deactivate System System->>System: Show failure message activate System System->>System: Log action deactivate System System->>System: Show failure message activate System System->>System: no connection to database deactivate System System->>System: deactivate System </pre>
<i>frequency</i>	often

For some tasks the automatic documentation that the task was completed using the care utility is not sufficient, e.g., if further relevant information is created and to be documented as well. An example is a discussion about the health status of a resident between staff

Table 3.9: Use case: Document further information

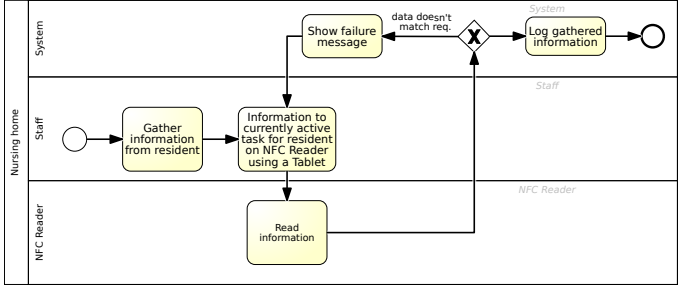
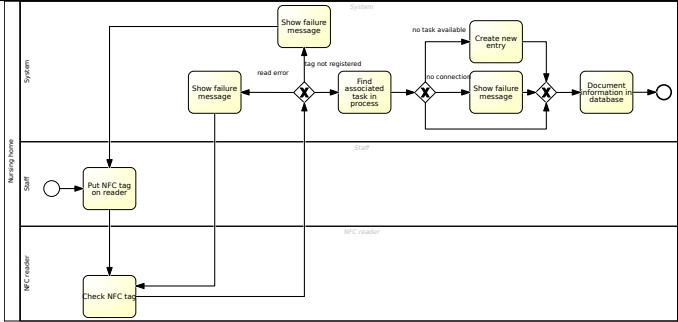
<i>primary actor</i>	care staff
<i>stakeholders and interests</i>	staff member (stakeholder 1) wants to document further information, e.g., the health status of a resident.
<i>preconditions</i>	additional information cannot be stored on NFC tag
<i>postconditions</i>	additional information is (correctly) stored in database (see <i>realization</i>)
<i>realization</i>	 <pre> sequenceDiagram participant Staff participant System participant NFCReader as NFC Reader Staff->>Staff: Gather information from resident Staff->>System: Information to currently active task for resident on NFC Reader using a Tablet System->>NFCReader: Read information NFCReader->>System: data doesn't match req. System->>Staff: Show failure message Staff->>System: Log gathered information System->>System: (End) </pre>
<i>frequency</i>	often

Table 3.10: Use case: Document injected task

<i>primary actor</i>	care staff
<i>stakeholders and interests</i>	staff member (stakeholder 1) wants to document his/her completed duties; nursing homes (potential stakeholder 2) wants documentation of higher quality
<i>preconditions</i>	NFC tag registered; NFC tag stores information; resident is registered
<i>postconditions</i>	task completion is stored in database (see <i>realization</i>)
<i>realization</i>	 <pre> sequenceDiagram participant Staff participant System participant NFCReader as NFC reader Staff->>Staff: Put NFC tag on reader Staff->>NFCReader: Check NFC tag NFCReader->>System: read error System->>Staff: Show failure message Staff->>System: Find associated task in process System->>System: no task available System->>Staff: Create new entry Staff->>System: Show failure message Staff->>System: Document information in database System->>System: (End) </pre>
<i>frequency</i>	rarely

member and resident. Use case 3.9 summarizes the necessary steps, i.e., the documentation of the additional information by the staff member using a form which is stored in the

Table 3.11: Use case: Delete resident

<i>primary actor</i>	staff of nursing home
<i>stakeholders and interests</i>	staff member (stakeholder 1) wants to maintain a current status in the database and to create room for new connections
<i>preconditions</i>	resident leaves nursing home; resident still connected to NFC reader
<i>postconditions</i>	NFC reader is disconnected and available for new connection (see <i>realization</i>)
<i>realization</i>	<pre> sequenceDiagram participant Staff participant NFCReader as NFC Reader participant System Staff->>NFCReader: Input Resident ID with Delete Request to NFC Reader NFCReader->>System: Read Data from NFC Tag System->>System: error when reading input System->>System: Show failure message System->>System: Remove entry in database and make device available System->>System: NFC already available System->>System: Show failure message System->>System: Abort task System->>System: End </pre>
<i>frequency</i>	rarely

system.

Use case 3.10 offers the option to document information connected to a NFC tag that is not assigned to any task. In this case, a task is injected and all information is documented in the database by the system.

Use case 3.11 describes the process if a resident leaves the nursing home and the connected NFC reading device (individual to this resident) is disconnected from the resident. The latter is realized by removing the associated entry in the database. Afterwards the NFC reading device can be reused, i.e., made available for a new resident.

3.3.4 Realizing automatic task completion in a process-aware care solution

The goal of this section is to show how automatic NFC-based task completion and documentation can be realized in a process-aware care solution.

Architecture

The process-aware care solution *ACaPlan*¹² has been developed for supporting the development and execution of flexible care processes based on established care guidelines, i.e., Nursing Interventions Classification (NIC), Nursing Outcomes Classification (NOC), and NANDA [KMS⁺15a]. *ACaPlan* builds on the multi-purpose cloud process execution engine CPEE¹³ [MRM14]. The user interface is realized through the *Who Cares* cockpit. Figure 3.10 depicts these existing components in black. In order to enable NFC-based task completion the existing solution is extended by the newly realized components depicted

¹²Adaptive Care Planning

¹³<http://cpee.org/>

with red, dotted outlines, i.e., (*Automatic Care Completion (ACC)*, *NFC Reader*, *XES Logging*, the data repositories, and optionally a *KPI Monitor*.

In order to derive a suitable set of components, the use cases presented in Sect. 3.3.3 have been analyzed for required common functionalities as well as data artefacts. This includes interaction with components and data artefacts, that are available through ACaPlan and the Process Engine. The results of this analysis are depicted in Fig. 3.9.

	Process Engine	ACC	NFC Reader	Hardware	Care Utilities	XES Logging	Patient Data	ACaPlan (e.g. TODO List)
Use case 1			×	×				
Use case 2			×	×			×	
Use case 3			×	×	×			
Use case 4			×	×	×			
Use case 5			×	×			×	×
Use case 6		×	×	×	×		×	
Use case 7		×	×	×			×	×
Use case 8	×	×	×	×	×	×	×	×
Use case 9			×	×			×	

Figure 3.9: Use case analysis: derivation of components and data artefacts

As can be seen in Fig. 3.9, each use case requires a set of functionalities and data artefacts. For example, in Use case 2, when a simple care utility is registered, the *Hardware* identification (tag id) has to be connected to an entry in the *Care Utilities* table. This also requires the *NFC Reader* component to scan the tag. In Use case 3 basically the same is done, except that more complex semantic information like side effects has to be saved. Also, for example for a pill tray, an individual NFC tag has to store dosage. The schema for the data that has to be stored on the NFC tag has to be saved in the care utilities data repository as well.

The granularity of individual components becomes clearer with each scenario. *NFC Reader* and *Hardware*, although present in all scenarios, are separate, because *NFC Reader* represents functionality, whereas *Hardware* constitutes a data artefact.

The necessity for the *ACC* components stems from the necessity to coordinate the information flow between different components when utilized from a process engine (see Fig. 3.11).

The final architecture is depicted in Fig. 3.10. Components for implementing the ideas in this approach are modeled with dotted outlines (red). All other components (black) are part of the existing solution the implementation is built on.

In the following, the general functionality of the newly realized components from an architectural point of view is described. At the end of this section the realization of the scenario “Give utility to resident” (cf. Table 3.8) illustrates the interaction between the components of the system.

The basis – Automatic Care Completion

Automatic Care Completion (ACC) as the main component has the following functionalities:

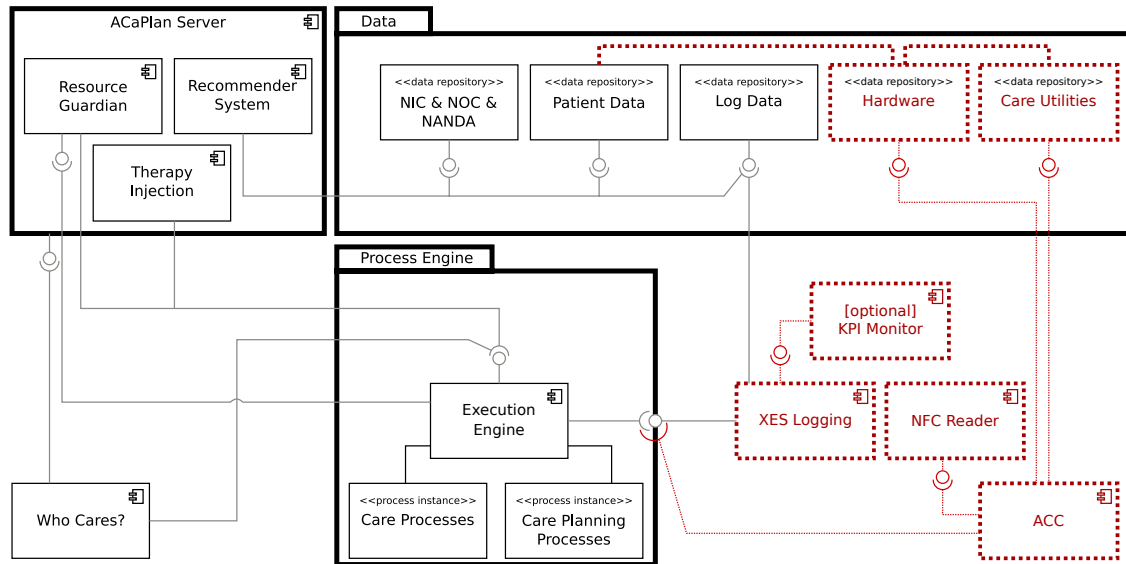


Figure 3.10: Architecture for process-based care system with embedded NFC-based documentation. Red-dotted components have been developed in this work.

- *Collection of events from the process engine implementing the therapy processes.* ACC knows all currently active tasks for all residents.
- *Collection of events from the NFC readers.* ACC knows all care utilities and the properties that have been used for each resident.
- *Completion of active tasks.* ACC selects all tasks that are potentially affected by a care utility and supplies it with information found on the NFC tag and allows the process engine to finish this task. Alternatively, it notifies the caretaker that additional (manual) input is necessary. In future revisions of the system, caretakers should be prompted to record voice messages which are automatically assigned to the correct task.

Data repositories: hardware & care utilities

In the hardware repository, all residents are stored with the ID of the related NFC reading device which is a prerequisite to relate events and residents.

The care utilities repository contains all data about the care utilities used during treatment, i.e., care utilities and their encoding, their type (simple or complex), interrelations with other care utilities such as side effects of drugs, and defined breaks between using the utilities (mainly relevant for drugs again).

Therapy processes often contain tasks that are executed in parallel. A resident, for example, has to be medicated and at the same time a proper level of hydration (periodic liquid intake) has to be maintained. While assigning the intake of a certain drug to a task might be unambiguous, proper hydration might be a side effect of different tasks such as serving the resident a cup of coffee or a glass of lemonade with the lunch, as well as the glass of water that is part of the medication intake all count towards the goal of proper hydration.

When hydration is seen as task in the therapy process that (in a loop) collects the quantities of liquid that has been consumed over the course of a certain time period, it becomes clear that whenever one of the above (separate) tasks happen, the hydration task additionally has to be provided with the correct information.

Thus, the care utility repository does not merely contain a list of data, but a flexible ontology that contains all facts connected to a certain care utility. Based on the ontology it becomes possible to identify the correct hydration task from a multitude of care utility applications. All facts in the ontology are described as turtle triplets and can be queried through SPARQL [PS⁺08].

Interaction Between the Components Based on a Scenario

As an example the implementation of the scenario conceptually described in Scenario 3.8 is provided. This scenario is chosen, because it covers a wide variety of interactions between ACC, nurses, residents, care utilities, and NFC reading devices (i.e. a full-stack example). In order to register the usage of a care utility (stored in “*«data repository» Care Utilities*”), the NFC tag attached to the utility has to be moved near the NFC reader.

In the “*«data repository» Hardware*” the IDs of the NFC readers for all residents are stored (thus the connection to the existing “*«data repository» Patient Data*”).

The component “*NFC Reader*” sends a stream of events to the “*ACC*” component, whenever care utilities are used with NFC readers.

The “*ACC*” utilizes the data repositories to interact with the “*Process Engine*” regarding the process execution. It is assumed that using a care utility always correlates to a currently active care task in the process engine. Every other situation is a possible violation of care standards.

However, the usage of care utilities also has a semantic side to it, that is not easily covered by imperative process models and their strict logic. For example, the glass of water coming with medication contributes to hydration (i.e., contribution to multiple tasks), or medication may have side-effects when given with other medication (i.e., detect problems in process logic). Of course, the process logic describing the care of a resident could encompass all fuzzy uses and side effects, but that would complicate the process model to a point where it would be hard to maintain by nurses.

Thus the “*«data repository» Care Utilities*” includes an ontology which is used by the “*ACC*” to generate warnings and identify the relevant active tasks. Finally, for complex care utilities the events include information such as dosage and time interval (e.g., for administering drugs) which again can be utilized to issue warnings and check compliance to care standards.

If “*ACC*” successfully identifies a care task, the data of the NFC tag (e.g. dosage information) will be sent to the process engine, and the event will be discarded (events are treated as a stream). The activity is then marked finished in the process engine, which in turn results in a log entry in the “*Logging*” component, and the next task will be marked as active. If “*ACC*” does not find a correlating care task, something not foreseen in the process happened. The “*ACC*” then registers the occurrence with the “*XES Logging*”, the progress of the care process remains unchanged.

The “*XES Logging*” component stores data as XML or YAML, containing timestamps of the events, the ID of a care utility with the described details, like inhibitors, suggestions and dosage, as well as notes and additional input from the caretakers. Additionally it provides

data to any interested external services, such as an optional ‘*KPI Monitor*’ component.

In order to realize and complete the solution design, a prototypical implementation supporting the scenarios presented in Sect. 3.3.3 is required. This section describes the design choices and implementation details in more detail (cmp. Fig. 3.10):

- Using ACaPlan and CPEE as the basic software stack.
- Using NFC on the hardware side.
- Using XES for logging.
- Process XES documents and KPIs.

ACaPlan & CPEE

The implementation extends an existing solution called Adaptive Care Planning (ACaPlan) [KMS⁺15a] that realizes therapy processes for nursing home. ACaPlan has been selected, because it is a solution in the domain that (a) has a modern message-based architecture and (b) is readily available as open source. ACaPlan provides a system, where a resident in a nursery is related to a therapy process. ACaPlan uses medical knowledge (derived from literature) which is stored in the “*«data repository» NIC & NOC & NANDA*”. A graphical user interface (*Who Cares*) to create and modify care processes for residents from this repository allows for simple exploration of examples and scenarios.

The implementation also utilizes event streams from the CPEE. The CPEE relies on a publish/subscribe model for a wide variety of events that occur during process execution. The approach relies on notifications whenever a task becomes active or is finished (per process, i.e., per resident), as discussed in Section 3.2.

The currently active tasks are compared to the events of by the “*NFC Reader*” component to determine which task (or which tasks) a certain care utility tackles.

It is important to note that the interaction with the Process Engine, ACaPlan, and concepts presented in this approach is generic. Thus, the CPEE used by ACaPlan could be replaced by other process engines, e.g., Apache ODE.

NFC

Background information on NFC is provided in Sect. 3.3.2. At the design level, whenever a NFC reader sends information about a care utility, with the help of the data repositories as depicted in Fig. 3.10 the following information can be deduced:

- Which resident is affected and thus: which care process is affected?
- What additional information about this care utility is available?

In order to make the functionality available, an open-source C library has been developed which talks to the NFC hardware¹⁴ via a hardware-supported binary communications protocol. This C library serves as a basis for high-level languages bindings.

¹⁴<http://www.kronegger.com/>

Interaction of Components

To give a more detailed view on this architecture, the sequence diagram in Fig. 3.11, shows the interaction of the new components in the system and how the execution engine gathers the data from the hardware repositories.

Figure 3.11 also explains how the ontology in the care utilities repository is working. For example, if the current active task asks for hydration and a glass of water is detected, a SPARQL query is used to understand the concepts of a glass of water and to see if the concepts of a class of water are actually a part of hydration care utilities.

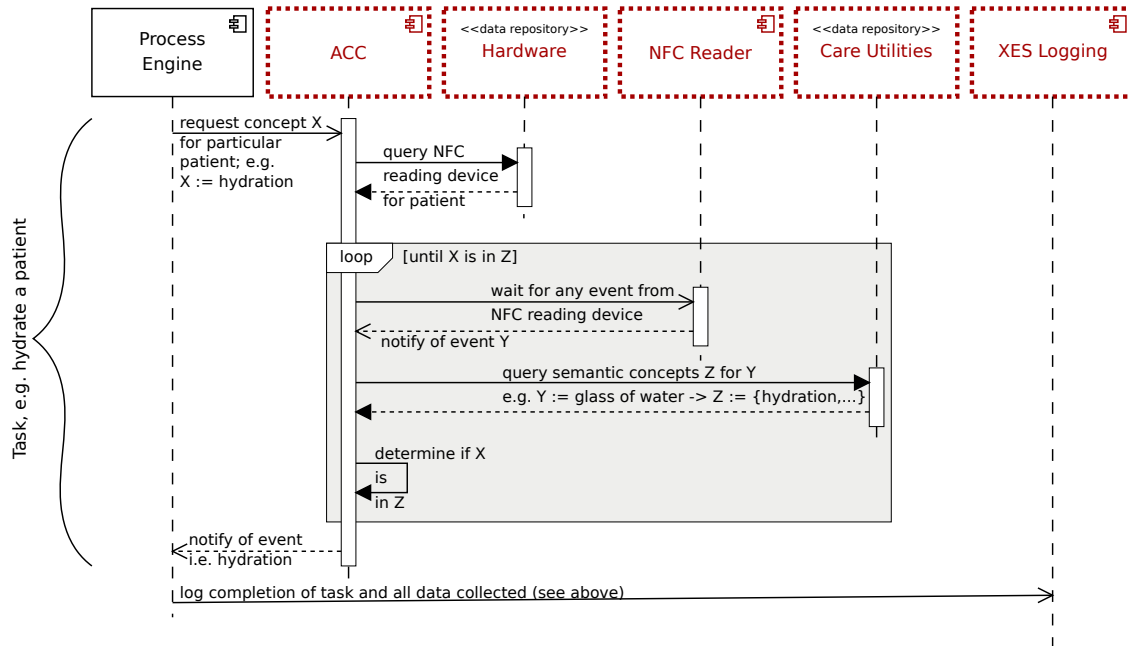


Figure 3.11: The communication of the components.

3.3.5 Practical evaluation

The evaluation is divided into two parts. At first, the reduction of documentation time and effort by automatic NFC-based task completion is evaluated with domain experts. Secondly, the feasibility of the creation of paper-based documentation from the XES logs is shown.

Reduction of documentation time and effort

The evaluation with domain experts is described based on its context, the data collection procedure, and the obtained results.

Evaluation context: The evaluation context is set by two different nursing homes. The daily routines in these nursing homes can be divided in morning routine, lunch, and afternoon & evening routine. Lunch mainly consists of two tasks, i.e., assist residents with intake and give drugs. For the intake a form has to be filled manually for documentation. Support for this documentation task has been outside the scope of this work so far. Giving

drugs is supported by the system. All of the process models have been modelled with experts from the nursing domain, specifically from the 2 nursing homes.

Morning routine The process model for the morning routine is depicted in Fig. 3.6 together with the assigned care utilities. One nurse is typically responsible for 14 residents. The morning shift for a nurse starts at 7:30 am and he/she has to conduct the morning routine all by himself/herself within 2 hours. Afterwards a second nurse joins. The tasks for the residents are split evenly. The residents have different levels of care, i.e., are able to conduct a varying number of tasks in the morning routine themselves. Even though the independence of the residents is to be preserved as much as possible, each of the tasks has to be documented by the nurse. This results in a higher amount of time for residents with a low level of care. As there is a substantial time pressure to finish the morning routine for all residents before breakfast, often the documentation of the morning routine is postponed to the end of the shift from noon to 13:00 pm. This constitutes a risk of lowering the documentation quality (e.g., by forgetting tasks).

Evening and Night Routine Figure 3.12 depicts the evening and night routine. For this shift a nurse is typically responsible for about 70 residents. During the night, additionally, a graduate nurse is present for emergency cases. The night shift runs from 8 pm to 7:30 am the next day. After handover of shift the general health status and clothes are checked. Repeating tasks of the night routine are changing incontinence pants and bed positions of the residents in order to avoid bedsores, typically performed every 2 to 3 hours. In quieter periods in between documentation is performed. In case of emergencies the nurse prepares the associated documents and calls the ambulance if necessary.

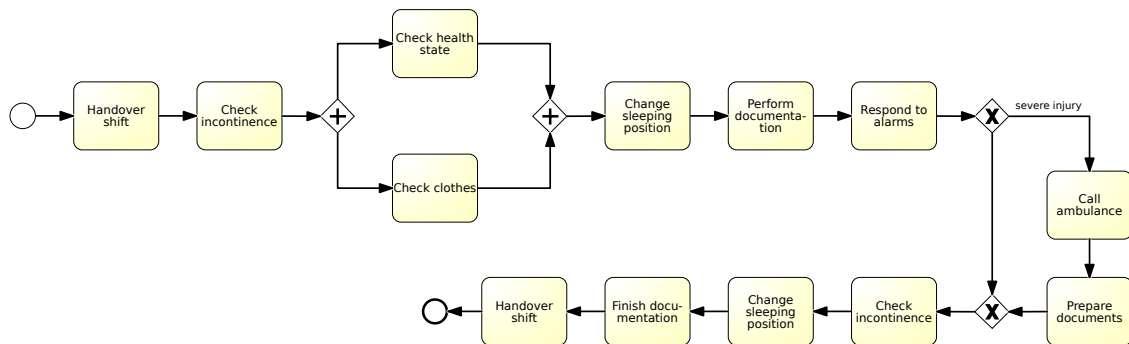


Figure 3.12: Process Model for Afternoon / Evening Routine

Data collection procedure:

The evaluation is based on two expert interviews with nurses from two different nursing homes. In the beginning the nurses were asked to describe the tasks that have to be done with every resident of the nursing home. This led to the process models, which can be seen in Fig. 3.6 and 3.12. Afterwards data has been collected on the average time spent on the documentation for each shift and when this documentation is typically done by asking them to log their behaviour for one week. At last, the nurses were asked, for each shift and for each task, how much time could be saved with the solution presented in this approach, taking into account that some tasks need additional information (i.e. comments

by the nurses) that can not be automatically acquired. Thus, the nurses, for each shift and each task, provided the following information:

- Potential physical objects and how practical it would be to use them for automatic logging.
- Current documentation effort, and estimated savings with the solution presented in this approach.

The time for using a physical care object is determined by three aspects:

- How long is the physical object used for the care purpose?
- How long does it take to bring the physical object next to a NFC reader (integrated into the bed)?
- How long has the physical object to remain next to a NFC reader, in order for the data to be read out?

While the duration of the first aspect is fixed, the two other aspects replace the manual documentation. They are assumed to take about 10 seconds in total per task¹⁵.

Additionally to the expert interviews, the whole system has been prototypically implemented in the lab of the research group. As can be seen in Fig. 3.7, a testing environment using nursing home beds and NFC reading devices has been built and the individual tasks have been replayed in the testing environment.

Obtained results:

Table 3.12 summarizes the results of the study which are explained in more detail in the following. The overall time reduction as expected by the experts are about 60%.

Table 3.12: Possible improvements through automatic documentation

Task	Morning		Evening		
Residents	14		70		
Tasks per resident	12		3-6 (4.5 avg.)		
Tasks total	168		370		
Worktime (nurses)	540 min.		420 min.		
Worktime per task	03:12 min.		01:06 min.		
Nurses	2		1		
Documentation	Manual	Automatic	Manual	Automatic	Improv.
per resident	07:30 min.	02:00 min.	03:00 min.	01:10 min.	68.00%
per nurse	52:30 min.	14:00 min.	210:00 min.	90:50 min.	64.87%
average per task	00:38 min.	00:10 min.	00:40 min.	00:10 min.	74.34%

Morning routine: 12 tasks have to be performed for 14 residents resulting in 168 tasks altogether. The average time per task is 03 : 12 min. The manual documentation time has been estimated as 07 : 30 min per resident, 52 : 30 min per nurse, and 00 : 38 min per task.

¹⁵It took about 2 seconds in tests with nurses, but we conservatively estimated 10 seconds to account for delays due to sloppy usage

Evening routine: Though the nurse is responsible for more residents the tasks are less intense and frequent. The shift handover takes about 30 minutes. On average, 370 tasks are performed by one nurse per night, taking 420 minutes of the shift. 210 minutes are left for documentation, resulting in an average documentation time of approx. 40 seconds per tasks. The experts were positive, that except for the health status check, all tasks contained in the process model in Fig. 3.12 can be documented automatically. Again, tests revealed a time of 10 seconds per task for automatic documentation under pessimistic assumptions. The documentation time per nurse will be reduced from 210 minutes to 90:50 minutes.

This means a reduction of 74.34% of time per documentation task, 68% per resident and 64.87% per nurse on average for both shifts. It should also be noted, that in this approach, the identity of the nurse fulfilling a task gets drawn out of the shift schedule. An adaption could be NFC tags placed on the clothes of a nurse as identifier.

Comprehensive documentation

The implementation is tested and forms are generated, following the guidelines of an Austrian nursing home, see Fig. 3.13. From the point of view of the nursing home there is no drawback or noticed difference regarding the operation without automatic documentation. One perceived advantage by the experts was, that when additional (available but not used) information needed to be included into the documentation, the documentation could be simply regenerated.

In order to evaluate if comprehensive documentation can be achieved a comparison of actual documents covering the morning routine of residents¹⁶ is done.

According to the involved care takers the created documents would be sufficient. As the care takers themselves consult the documentation of fellow care takers, in order to learn details of the residents, they anticipate automatically generated documents to be easier to read, as they all strictly follow the same structure and granularity.

In Fig. 3.13, a typical paper-based documentation can be seen, while Fig. 3.14 shows the equivalent in XES.

In the following the process on how the data was processed to generate the paper-based documentation is elaborated. ① shows the timestamp of the documentation. In the log, the timestamp exists for each task. For the paper-based documentation just the date, not the time is necessary. ② shows the name of the nurse. The fact that multiple nurses may share the workload for one resident, is reflected in the paper-based documentation. In the log the responsible actor is again saved with each task. ② documents the behavior of a resident in the community room. In the log that this is a note attached to a certain task. For the current implementation nurses can take audio notes with their smartphone which are automatically converted to text and saved with the last task that was worked on (i.e. activated through a care utility). ③-⑧ show care tasks as defined in Fig. 3.6. The text is generated automatically from (1) the existence of the log entry and (2) notes attached to the log entry, as can be for example seen with ⑥a and ⑥b. Tasks ④, ⑤, and ⑦ have been omitted in the log to simplify presentation, but look similar to ③. ③+④ in the paper-based documentation are a special case, as the standardized sentence “*The morning hygiene has been carried out*” depends on the existence of multiple tasks.

¹⁶No personal data that allows to infer on residents is included in the data excerpt presented below

3.3 Creating Accurate Data Sets for Process Mining with Human Resources Involved

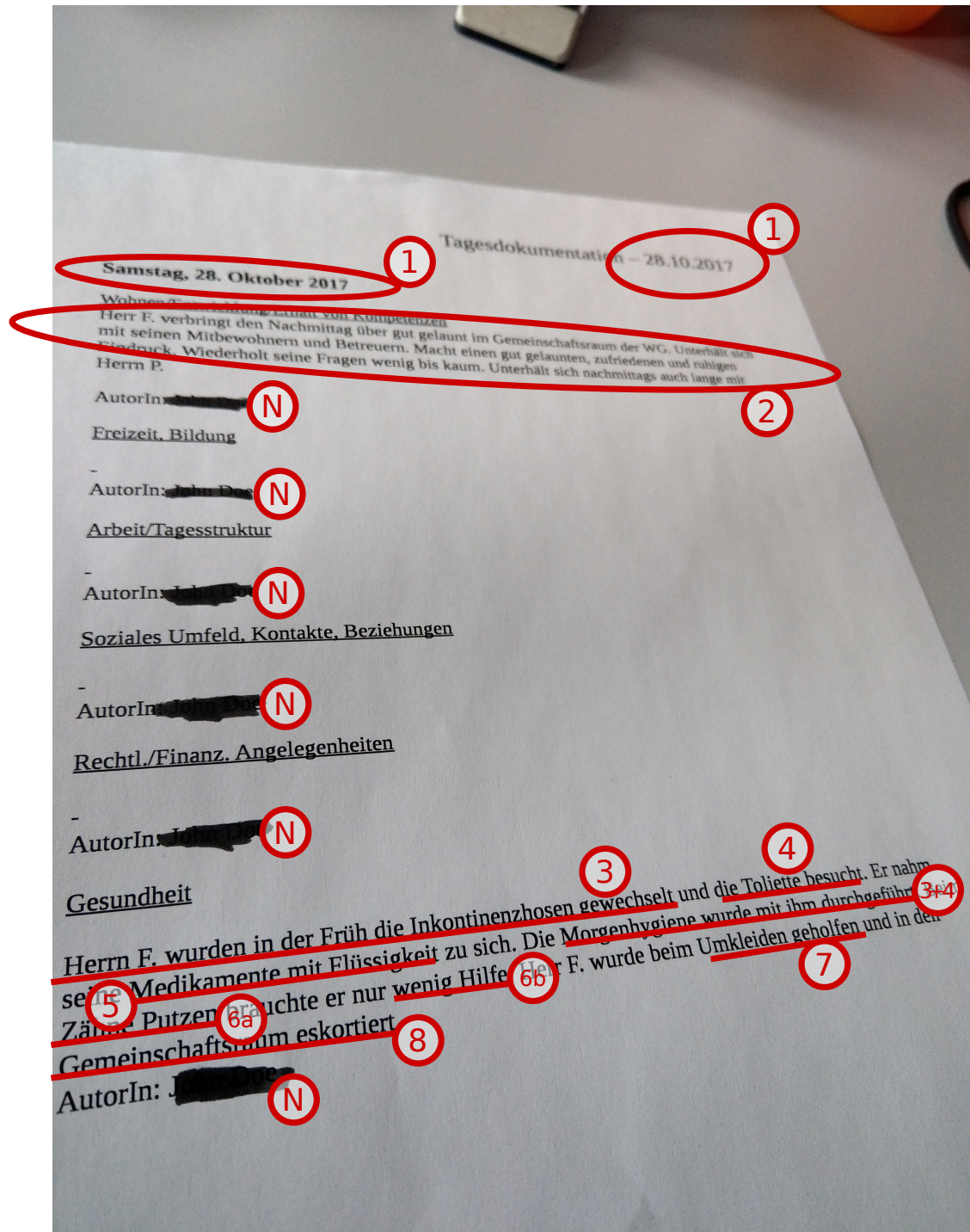


Figure 3.13: Paper-based documentation

The newly gained information about the exact point of time an event occurred due to the reading of the NFC tag, can be used to analyze the care process for each resident or of the nurses even, for example, the intervals between intakes of medication.

```

<trace>
  <string key="concept:name" value="Resident F."/>
  <event>
    <string key="concept:name" value="Change incontinence Pants"/>
    <string key="org:resource" value="John Doe"/>
    <date key="time:timestamp" value="2017-10-28T06:32:00.000+01:00"/>
  </event>
  ...
  <event>
    <string key="concept:name" value="Brush teeth"/>
    <string key="org:resource" value="John Doe"/>
    <date key="time:timestamp" value="2017-10-28T06:42:00.000+01:00"/>
    <list key="data_received">
      <string key="content" value="
        Wenig Hilfe.
      "/>
    </list>
  </event>
  ...
  <event>
    <string key="concept:name" value="Escort to community room"/>
    <string key="org:resource" value="John Doe"/>
    <date key="time:timestamp" value="2017-10-28T06:48:00.000+01:00"/>
    <list key="data_received">
      <string key="content" value="
        Herr F. verbringt den Nachmittag über gut gelaunt im
        Gemeinschaftsraum der WG. Unterhält sich mit seinen Mitbewohnern und
        Betreuern. Macht einen gut gelaunten, zufriedenen und ruhigen
        Eindruck. Wiederholt seine Fragen wenig bis kaum. Unterhält sich
        nachmittags auch lange mit Herrn P.
      "/>
    </list>
  </event>
</trace>

```

Figure 3.14: XES log (log-based documentation)

3.3.6 Discussion

This section reflects on different aspects that are crucial for the success of automatic task completion and documentation as proposed in this approach. At first, the applicability of the approach in the care domain is discussed (cf. Sect. 3.3.6). The transferability of the approach to other domains such as production are discussed in Sect. 3.3.6.

Estimation of Cost Categories

The evaluation showed that potentially a good amount of time could be saved during tasks of documentation. If time can be saved, other aspects are affected as well. In [JVKN08], the four dimensions, time, quality, cost, flexibility are shown to in relation to each other. In the view of performance measurement, the following points can be established.

Time Dimension In the time dimension, a look at the lead time is taken. The lead time is defined as the time it takes to finish a whole case. A case in this scenario would be the complete morning shift. With the reduction of time spent on documenting tasks, the lead time can be reduced.

Quality Dimension The approach focused on the internal quality in the quality dimension. The NFC service can provide immediate feedback to the nurses and the documentation is done in an automatic way.

Cost Dimension The cost dimension is directly connected to the time and quality dimensions. With a decrease in the lead time, costs can be reduced. Moreover, a documentation with a poor quality can increase costs, if steps have to be redone or unsatisfied residents file in complaints.

The flexibility dimension is omitted in this discussion, since this dimension is untouched from this approach.

While the chosen example on a nursing home, contains only a small number of residents, this solution can easily scale to bigger environments like a hospital. In a hospital the number of persons needing treatment would increase greatly, but also the nursing staff and the amount of hardware. The price of this solution should increase linearly, because a one to one relation of NFC reading devices to the number of beds is established. The number of care utilities used, increases as well linearly, since the same care utilities are used for each person.

Transferability to other domains

The evaluation indicates the potential for integrating automatic NFC-based task completion into process-aware care solutions. In the following, other potential domains are discussed, which might benefit from automatic NFC-based task completion as well.

At first, domains can benefit that possess similar characteristics as the care domain. In [KRVM14, KR17], several of these domains are identified such as manufacturing and event management. In all these domains, process-aware solutions are expedient. Process instances are connected to different subject, e.g., the patient, the product, or the customer. Process data as well as environmental data plays an important role as well as different actors working on the tasks. Some of these domains such as manufacturing (but also, for example, logistics) are already familiar with the use of sensors. Documentation plays an important role in these domains – for different reasons – as well. Some also are subject to documentation obligations and for some the monitoring of KPIs is daily business. Finally, in all these domains, physical objects are employed in process tasks such as machines in production and goods in logistics. In summary, the factors for a domain to potentially benefit from automatic NFC-based task completion are:

1. Process-aware solution
2. Need for documentation/KPI monitoring
3. Process subjects and / or objects can be equipped with NFC tags and connected to NFC reading devices. A process subject [GR15] denotes the person or item that denominates the process instance, e.g., the resident or the product. Process objects describe in a broader sense data that is processed during process execution as well as physical objects that are utilized for conducting process tasks such as a vehicle, a comb, or an employee card.

Application areas that fulfill the above-mentioned preconditions are, for example, manufacturing and logistics. Both crave for process support [SSSA12, BCD15] and are prone to documentation for quality assurance and traceability [GSC09]. Specifically in the logistics domain, sensor-based technology such as RFID is already in use [CCL07]. Moreover, manufacturing and logistics processes employ process subjects, i.e., products and goods/cargo

as well as process objects such as vehicles and machines that can all be equipped with NFC technology.

Manufacturing: The applicability of sensor-based documentation in the manufacturing domain was analyzed in the experimental manufacturing environment LegoFactory¹⁷, at WST research lab. In this setting, several sensors are integrated and utilized anyway. Here the product is the driving factor for the process execution, i.e., the product is to be equipped with a NFC tag and the different machines with readers in order to document automatically that a product has passed a certain machine.

Logistics: Similar to the manufacturing domain, the goods are the process subjects which drive the process execution. Hence, for logistics as already done in practice, goods can be equipped with NFC tags and the utilities for transportation, e.g., the truck, equipped with the readers. This would not only facilitate documentation, but also foster the traceability of the goods on the transport. An interesting question is whether single goods are equipped with NFC tags each or cargo, i.e., bundles of goods. This becomes particularly important for bundling and unbundling of cargo.

Currently, only a restricted amount of data input can be processed through NFC technology, e.g., dosage. For future applications, extended solutions connecting automatic documentation with data input are conceivable as well.

Limitations

- The process-aware care solution has been developed using the information from nurses prototypically implemented in a lab setting, i.e., It has not been tested in a real-world environment yet. A process-aware system environment would be beneficial to implement this system, although a data centric approach can be used as well. The disadvantage of not having a process-aware solution, is that information of the NFC tags can only be documented without the enriched information of the ontology and the information of the whole care process. So instead of a glass of water has been given to achieve the task of hydration, only a glass of water would be documented.
- Technical knowledge is required for writing information on an NFC tag, which imposes a burden. This problem can be mitigated with the help of an easy to use software. However, a new task concerning technical interactions is still added to the daily routine.
- If a tag is faulty or contains wrong information, the information has to be rewritten on the tag, even though the medical container could contain the correct medication.

3.4 Conclusion and outlook

This chapter focused on the creation of a data set, which can be used for process mining algorithms. It is crucial, that the data sets should be able to be generated during the execution of a process, thus the file format needs to be modifiable in a short amount of time. In addition, these data sets should be generated automatically without any required transformation algorithms. The input of human resources needs to be automatically processed as well.

¹⁷<http://gruppe.wst.univie.ac.at/projects/LegoIndustry/index.php?t=project>

To summarize the main contributions of this chapter:

- Different formats for process mining data sets have been analyzed and a new XES serialization has been presented in Section 3.1. YAML has been chosen, since it can handle the same amount of information as XML, yet also is modifiable in a shorter amount of time.
- An approach has been described and evaluated in Section 3.2, which allows to log process instances directly during their execution. These data sets can be used as an input source for process mining algorithms, on- and offline.
- The human interaction with a process execution engine has been studied in Section 3.3, enabling humans complete activities in a process instance and automatically document their activities, thus creating process mining data sets with human involvement during the execution of process instances.

The next chapter, Chapter 4, introduces novel process mining algorithms, that focuses on the detection and determination of concept drifts of a process.

4 Discovering the Evolution of Processes through Concept Drifts

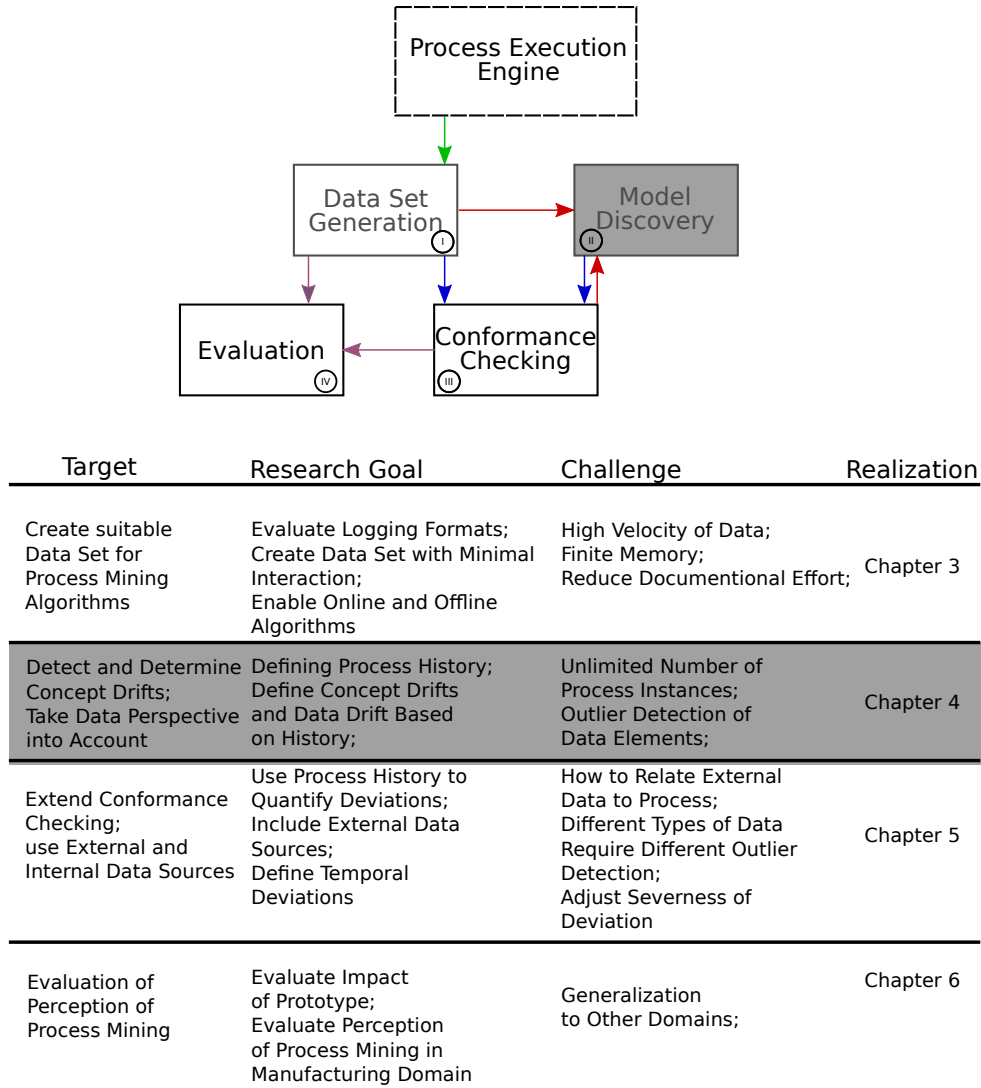


Figure 4.1: Overview of Targets, Research Goals, Problems of this thesis. The features for this chapter are marked.

This chapter addresses objective ② in Fig. 4.1, i.e., the discovery of the evolution of a process by detecting and defining concept drifts.

A process typically changes over time [RW12], for example due to, a new legislature that forces adaptations in a process, e.g., the introduction of a new law, enforces medical staff

members in a hospital to conduct a special test for a specific disease on patients before they are admitted to the hospital. Another common reason for a change in a process are environmental factors, i.e., the temperature or the weather. Custom labels, for example, cannot be printed on champagne bottles in a factory when at a high temperature. Thus this option is not available in the summer. These changes are often present in the process logic, but not always updated in the current process model, which creates a mismatch between the execution of process instances and the process model. These changes are **concept drifts** [BVDAZP14]. It is important to note, that such drifts can be occurring on the control flow level, but also at the data level, i.e., data elements attached to the events. It is of utmost importance to identify these drifts as soon as possible and to remedy the probable mismatch between process logic and process model. To discovery and detect all concept drifts of a process, the following research questions are derived:

- RQ ②a How can the evolution of a business process be discovered at runtime?
- RQ ②b How do data elements relate to concept drifts?

Section 4.1 tackles RQ ②a. It introduces the concept of **process history**. Using an event stream as an input source, a process model is detected. Every time a certain number of process instances are not fitting the current process model, a new process model is discovered and the previous model is saved in a collection of process models for this process. This signals a concept drift at the control flow perspective. In addition to the collection of the process models, Section 4.1 provides a formal definition for the four types of concept drifts based on the process history, e.g., incremental drift, recurring drift, gradual drift and sudden drift [BVDAZP14]. The algorithms to discover and identify drifts are evaluated on an artificial data set.

Section 4.2 focuses on RQ ②b, drifts at the data perspective. A drill, for example, uses a certain amount of power which is stored as a data element in an event. If the drill is becoming dull, the amount of power which is used for a work piece increases. This change can be detected as a drift at the data perspective and with the support of process history, the drill head can be changed before unwanted accidents occur. Thus Section 4.2 provides a formal definition for all four drifts, previously described on the control flow perspective, at the data perspective. A drift in a data element can be detected, by identifying outliers, tailored to the type of data element, i.e., a numerical value requires a different outlier detection than an arbitrary string. Algorithms to discover outliers and identify drifts are evaluated on a real-world data set from the manufacturing domain.

A selection of text, figures and tables within this chapter is based on the following publications.

Stertz, F., Rinderle-Ma S.: Process histories-detecting and representing concept drifts based on event streams. In: On the Move to Meaningful Internet Systems. OTM 2018 Conferences - Confederated International Conferences (CoopIS), Pages: 318-335
https://dx.doi.org/10.1007/978-3-030-02610-3_18

Stertz, F., Rinderle-Ma S.: Detecting and Identifying Data Drifts in Process Event Streams Based on Process Histories. In: CAiSE Forum 2019, Pages 240-252, https://dx.doi.org/10.1007/978-3-030-21297-1_21

4.1 Concept Drifts at the Control Flow Perspective

Business processes have to constantly adapt in order to react to changes [RW12] induced by, for example, new regulations or customer needs often lead to concept drifts [BVDAZP14], since not conforming to a new legislation, can cause tremendous fines [Dro17]. Process changes are explicitly defined and stored in change logs [RRJK06] and hence are known to the company. Contrary, concept drifts are happening as the process evolves and hence are to be detected from process execution logs, i.e., logs that store events of executing process instances such as starting or completing process tasks. By now techniques to detect concept drifts in business processes work on process execution logs and are hence applied ex post, i.e., after the process is finished. However, detecting concept drifts during run-time bears many benefits such as being able to instantly react to the concept drift.

Run-time detection of concepts drifts works on event streams rather than on process execution logs. As event streams are infinite, online concept drift detection faces the following challenges:

- The start and end event of the stream are unknown due to the infinite nature of the stream.
- It is not known how many events belong to a trace since new events can still appear in the stream.
- it is not known which future events will occur and when they will occur.

Moreover, different kinds of concept drift are to be distinguished, i.e., incremental, sudden, recurring, and gradual drifts [BVDAZP14]. So far, the focus has been put on incremental and sudden drifts only. However, detecting recurring and gradual drifts can be important for many application domains as well.

To detect the evolution of a process, *process histories* are introduced. A process history reflects models that are discovered for a process based on an event stream. Process histories provide a novel way to detect and represent concept drifts through mining the evolution of a process model based on an event stream. The challenging question is when a new model is created, i.e., which event or sequence of incoming events triggers the creation of a new model in the history. Two new algorithms are presented in this section. The first algorithm creates the process history and discovers new models. The detection of a new model, is based on conformance [RVdA08] and the “age” of the event information using the sliding window approach, i.e., older process instances do have no impact on the current business process logic. The second algorithm determines concept drifts based on the synthesized process histories and enables the detection of incremental, sudden, recurring, and gradual drifts. The evaluation comprises a prototypical implementation as well as a comparison with existing approaches on detecting concept drifts based on synthetic logs. In summary, this section provides means to detect incremental, sudden, recurring, and gradual concept drifts based on event streams and the concept of process histories during run-time.

In Section 4.1.1, a formal definition of a process history is provided and the different types of concept drifts explained. Section 4.1.2 features the algorithms to synthesize a process history and determine the type of the concept drift. An evaluation is present in Section 4.1.3, based on a synthetic log created using the process models of [BvdAŽP11].

4.1.1 Fundamentals of Process Histories

This section defines process histories and the necessary data structures for the proposed algorithms. The different types of concept drifts are described at the end.

Process History. A process history reflects a collection of process models that have been executed for a majority of process instances at a certain point in time, i.e., a collection of all process models between all concept drifts. It is defined as follows:

Definition 1 (Process History). *Let P be a business process. A process history H_P is a list of process models $M_n, n \in \mathbb{N}$ that have been discovered for P . M_n is the current model for P . A new model is discovered, if the certain amount of currently active process instances are not fitting the current model.*

$$H_P := \langle M_0, M_1, \dots, M_{n-1}, M_n, \dots \rangle \quad (4.1)$$

For synthesizing a process history, process model discovery and process conformance checking algorithms are applied in Alg. 1 in Section 4.1.2 to find, check and adapt the current process model. A small example can be seen in Figure 4.2. Both activities in this process model, are executed in sequence. Suddenly, in new process instances “Abrosia” is executed sometimes before “Antispasmodic Drugs”, but not always. A new process model is detected, as can be seen in Figure 4.3. This figure presents a similar process model, where the same activities are executed in parallel and it fits the new process instances. A concept drift happened and the process model in Figure 4.3 is now the current model for this process.

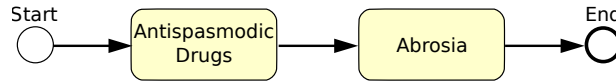


Figure 4.2: Small process model example based on the running example in Figure 1.1. The activities are executed in sequence.

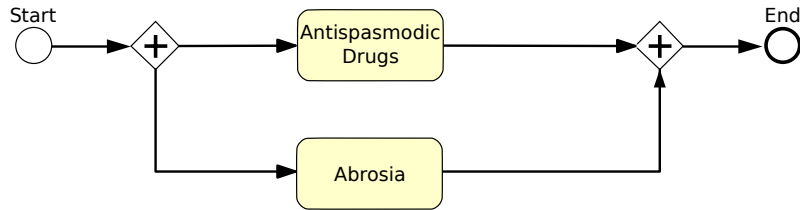


Figure 4.3: Small process model example based on the running example in Figure 1.1. The activities are executed in parallel.

For the implementation of Alg. 1, specific data structures are required to keep track of already handled events and create identify their traces in order to discover a process model, described in the following.

Data structures: A process history covers all process models that have been executed for a specific process. Any time a task is executed, an event is created and injected into an event stream.

Here, an adapted version of stream-based abstract representation (S-BAR) [vZvDvdA18] is used, since it can be applied directly onto an event stream. S-BAR introduces an abstract representation of the directly follows set of events. This set of events consists of every observed pair of subsequently executed events, created by using two maps. In this case, a map relates to the well-known data structure of a *hash table* [CLRS09], consisting of keys and their corresponding values.

The **trace_map**, is built using the trace id of a trace as key. The corresponding value to a trace id, is the whole trace. In an event stream, one event at a time is processed. After processing the event, is put into the trace_map. To cope with memory issues and to help determine active traces, the point in time when the first and currently last event of a trace is being processed is also stored. The second map, **directly_follows_map** represents the directly follow relations of all events w.r.t. the trace_map. As key, the preceding event is being stored, with the following event as the corresponding value.

The usage of these maps is explained in detail in Section 4.1.2. Figure 4.4, shows the trace_map and directly_follows_map for the traces [A,B,C] and [A,C,B]

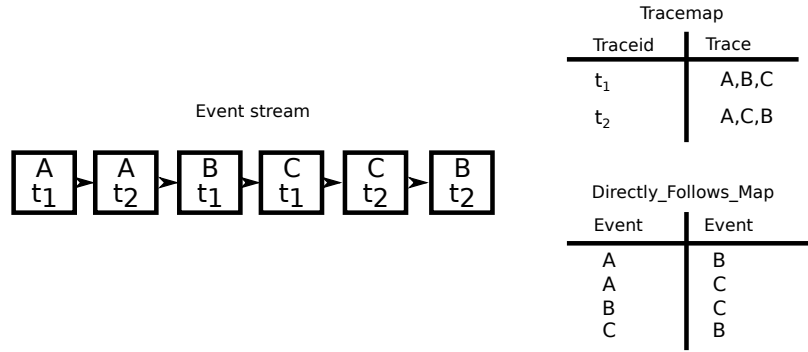


Figure 4.4: Event stream containing two traces with different order of events.

This approach is using the inductive miner, since it always generates a sound process model. For using the inductive miner, an assumption of specific start and end events is required. Since an event stream is used as an input source for the algorithms, it cannot be guaranteed to identify the correct end or start events. The approach is constructing a set of potential start events, by collecting the start events of each trace discovered in the event stream. The example in Fig. 4.4, shows two traces that both have the same starting event “A”, so this event would be the only start event for the inductive miner. For the end events, only the last known events of already known traces can be considered. The traces in Fig. 4.4 provide two possible end events. The first trace has as an end event “C”, while the second trace has as an end event “B”. This results in a set containing two end events marked for the inductive miner, namely “B” and “C”.

Sliding Window: The two maps, trace_map and directly_follows_map, contain the necessary information about every processed event, i.e., point of time of execution, data elements of events, e.g., the name of an event and the corresponding trace. Since business processes change, already finished or older traces may be part of a preceding version of the business process. Not every trace can be saved in the main memory, because of capacity issues. To resolve that, the sliding window approach is used. The sliding window

only stores k data entries, here keys in the `trace_map`. If there are already k entries stored, the oldest one is removed before storing the new entry. If the `trace_map` would be exceeding k , the key value pair with the oldest currently known end event is removed. This method ensures, that only currently active and newer traces are taken into account while discovering a new process model and checking its fitness.

Concept Drifts reflect a shift in the business process logic, meaning that the execution of a business process changed over time. There are several reasons for a change in the process model, like a new business policy or adaptations in the business process logic to meet customer needs. Every time the business process logic changes, a new process model is discovered.

[BvdAŽP11] describes 4 kinds of types of concept drifts.

- *Sudden Drift*. It shows a complete new workflow for the business process, for example caused by a new legislation, like GDPR.
- *Recurring Drift*. There could be a process model that is used for a specific time in the year, for example, Christmas season, in which workflows are executed differently to meet customer needs. These drifts appear periodically and replace the process model with another already known process model.
- *Incremental Drift*. Describes small changes, that are natural in the evolution of a business process. Especially in the beginning, the process history will be often extended, because a new process model is discovered after each new event in the event stream. This results in many sub process models.
- *Gradual Drift*. The process got changed and all process instances since the change point have a different process model. M_n and M_{n-1} coexist, as long as already started process instances of M_{n-1} are still running.

A closer look at concept drifts, can be seen in Chapter 2. It is to be noted, that recurring drifts and incremental drifts, can also be gradual drifts, since process instances of the PM_{n-1} could still be executed.

A process history enables the detection of each of these drifts and is defined in the next section.

4.1.2 Algorithms for Synthesizing Process Histories

This section describes the synthesizing process histories and detection of concept drifts.

Synthesizing a Process History: For detecting the evolution of a process, a process history, H_P , is synthesized. The process history contains a list of already known process models, M_i , where M_0 is the first known process model for P and M_n is the last known and currently used model for P . With the list of process models, all historical changes of P 's logic are represented in H_P , and show the evolution of P .

The developed Algorithm 1 synthesizes a process history based on an event stream and is described in the remainder of this subsection. As input an event stream, ES , a window limit k and the thresholds ϕ and σ are required. The thresholds are described in detail in the following paragraphs.

At the beginning the process history, H_P , is an empty list and does not contain any process models. The `trace_map`, explained in Section 4.1.1, is also empty and contains

no items in the beginning. The `directly_follows_map` is created after an unfit trace is detected based on the content of the `trace_map`.

The sliding window approach is used, to only take recent process instances into account, since the approach focuses on detecting new process models and older instances are removed from memory. For usage of the sliding window approach, the window size k must be defined. Only k items are possible in the `trace_map`. Every time a new event is processed, it is checked, if its trace id is already existing in the `trace_map`. If it does not exist and the size of the map is smaller than k , the trace id is used as key and as a value, the event is used as the starting event of the corresponding trace. If the map has already k items, the oldest trace is removed from the `trace_map`. If the trace id is found in the `trace_map`, this event will be appended to the trace.

Afterwards, if there is already at least one model in the process history, the fitness of the active trace is checked. The fitness of a trace reflects the conformance of a process instance and is defined between 0 and 1, where 0 reflects non-conforming instances and 1 conforming instances. For this purpose, common conformance checking algorithms are used. Conformance checking algorithms are explained in detail in Chapter 2. In short, the event sequence of a process instance is aligned to a process model and for every deviation the fitness is decreased and a move is detected. For the alignment costs of the trace, two costs are calculated. The costs for a move in a log, describe if an event is found in the log but not in the model at this position. Costs for a move in a model, describe if an event is found in the model but not in the log. For our purposes, only moves in a log are considered, because in an online environment, it is not known, if a process instance has reached its end event yet, which means, that the trace can still fit the model.

Assume that the model in Fig. 4.5, is our last known model in the process history. The two traces that perfectly match are $[A, B, C, D]$ and $[A, C, B, D]$. Since only the moves in the log are taken into account, the two traces $[A, B, C]$ and $[A, C, B]$ receive a perfect score, and it is assumed, that those process instances are still being executed and the end event “D” has not been processed at the moment.

The last trace $[A, D, B]$, received a lower fitness score, based only on moves in the log. The second event “D” is not expected this early in the process model and cannot be aligned in a perfect way, so it is moved in the log.

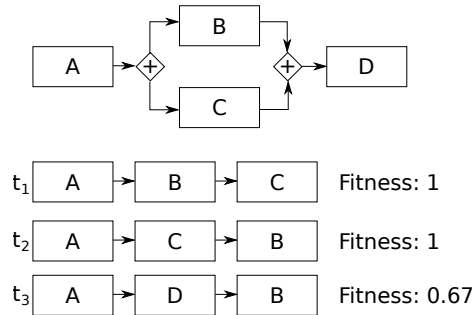


Figure 4.5: A process model with one parallel gateway and 3 related traces. While the Move-Log fitness is perfect for the first two traces, the last trace contains an additional event and receives a lower score

To define if a trace fits the model, a threshold, σ is introduced, ranging from 0 to 1. While 0, would result in any trace fitting any model, 1 would only consider perfectly

Input: Event Stream ES (a series of events)
 k (Limit for number of trace_map items)
 σ (Threshold for the fitness of a trace for a model, [0,1])
 ϕ (Threshold for distinction of a new viable model [0,1])

Result: Process History H_P (contains all viable process models in chronological order.)

```

 $H_P = []$ 
 $M\_duration = 0$ 
trace_map<trace_id,trace> = 0
for  $e$  in ES do
  if trace_map contains_key  $e.trace\_id$  then
    | trace_map[ $e.trace\_id$ ].append( $e$ )
  else
    if trace_map.size  $\geq k$  then
      | trace_map.delete_oldest
    end
    | trace_map.insert( $e.trace\_id, e$ )
  end
  if  $H\_size \neq 0$  and conformance_checking(traces[ $e.trace\_id$ ],  $H_P.last$ )  $< \sigma$  then
    directly_follows_map<event,event> = 0
    for  $t$  in trace_map.values do
      if conformance_checking( $t, H_P.last$ )  $< \sigma$  then
        for  $i$  in  $t.size$  do
          if  $i \neq 0$  then
            | directly_follows_map.insert( $t[i-1], t[i]$ )
          end
        end
      end
    end
    Model = inductive_miner(directly_follows_map)
    fitting_traces_counter = 0
    durations = []
    for  $t$  in trace_map.values do
      if conformance_checking( $t, Model$ )  $\geq \sigma$  then
        fitting_traces_counter += 1
        if  $t.end\_event$  in Model.end_events then
          | durations.append( $t.end\_event.time - t.start\_event.time$ )
        end
      end
    end
    ScoreModel, trace_map.values = fitting_traces_new / trace_map.values.size
    if  $s \geq \phi$  then
       $H_P.append(Model)$ 
       $M\_duration = durations.average + durations.std\_deviation$ 
      unfinished_traces =
        trace_map.get_unfinished( $H_P[H_P.size-1], trace\_map, M\_duration$ )
      detect_concept_drift(trace_map.values, unfinished_traces,  $H_P, \phi, \delta$ )
    end
  end
  if  $|H_P| = 0$  then
    |  $H_P.append(inductive\_miner(e))$ 
  end
end

```

74

Algorithm Part 1: Algorithm for synthesizing a process history based on an event stream.

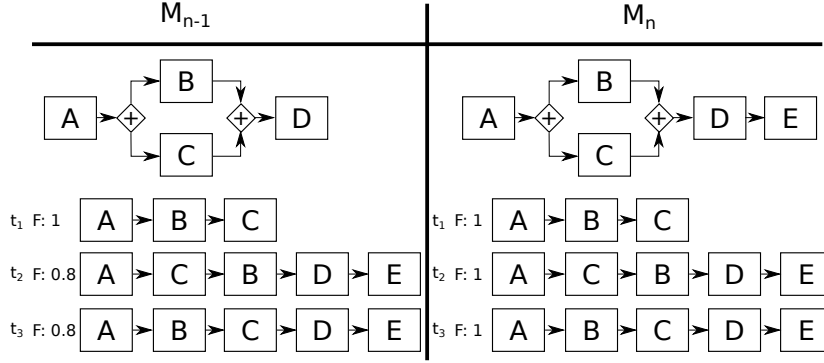


Figure 4.6: Model M_{n-1} is only fitting the first trace perfectly. Model M_n is fitting all traces. M_n is now the new model.

matching traces as fitting. For the purpose of detecting viable process models, a high threshold like 1 is suggested. This guarantees to only consider perfectly matching traces for the distinction.

If the trace of the currently processed event, does not fit the last known process model of the process history, a new model is mined, using the inductive miner. As input for the inductive miner, the abstract representation of the directly follows relation is sufficient. Only unfitting traces in the current window are used for discovering the new process model. The inductive miner always produces sound workflow nets and suffers from less instabilities like the α -miner, which does not detect short loops for example.

To distinguish between viable new process models and anomalous process instances, a score for a process model for a set of traces is defined as:

Definition 2 (Model score). *Let T be a given set of traces, M a process model, and $\phi \in (0,1]$ be a threshold. Moreover let $A \subseteq T$ be the set of all traces having a fitness score greater or equal than ϕ and let $\chi_A(t)$ be the indicator function, returning 1 if $t \in T$ is in A , 0 otherwise.*

Then $\mathcal{S}_{M,T}$, the model score of process model M w.r.t. T , is defined as

$$\mathcal{S}_{M,T} = \frac{\sum_{t \in T} \chi_A(t)}{|T|}.$$

In Algorithm 1, for calculating the score for the new model using the current trace_map, the values of the whole map, are checked for conformance with the newly discovered process model. If a trace is fitting the new model, a counter is increased by 1, starting at 0. In addition, if the trace's end event is one of the end events of the new model, the complete execution time is calculated and stored in a list of execution times for this model. The variable $M_duration$ describes the average execution time of M plus the standard deviation. The number of fitting traces is then divided by the number of all possible traces from the trace_map, which results in the score for the new model and the trace_map. The score for this model, ranges from 0 to 1 as well. To determine if the new model is viable, the score must be greater or equal than ϕ . ϕ is introduced as a threshold between 0 and 1, where 0 considers any model as viable and 1 only models that fit every trace from the current window to the new model. For the history in the evaluation, only models fitting at least 90% of the traces from the current window have been considered to get a strict list of viable models, with results discussed in Section 4.1.3.

In Figure 4.6, the detection and creation of a new process model in the process history is shown. On the left, the two longer traces do not fit the last known process model in the process history. The newly discovered model, visible in Fig. 4.6 on the right, is able to fit all current traces into the model. Since the new model fits all current traces, the new model is appended to the process history. The old model is now M_{n-1} in the process history and the newly created model is now the last and current model in our process history, M_n .

Every time a new model is discovered and appended to H_P , a concept drift is detected. To determine the type of the concept drift, unfinished traces for M_{n-1} from the `trace_map` need to be collected. A trace is likely to be unfinished if its end event is not part of the end events of M_{n-1} and its current execution time is lower than the execution time stored in $M_duration$. If its execution time is larger, the process instance is likely to be canceled.

Concept Drift Distinction: Algorithm 1 synthesizes a process history for a specific process. Every time a new process model is appended to the process history, a concept drift is detected. The 4 types of concept drifts, in relation to a process history, can be defined formally as follows:

Definition 3 (Concept Drift Types). *Let T be a given set of traces and U be a given set of unfinished traces. Moreover let H be a process history for a process P and $\delta \in [0, 1]$, $\epsilon \in [0, 1]$ be thresholds and the function *fitness*, defined for one trace and a model, ranging from 0 to 1. The following drift types are defined as follows:*

- *Incremental Drift if $|H| \geq 2 \wedge \exists(t \in T, \text{fitness}(t, M_{n-1}) \geq \delta \wedge \text{fitness}(t, M_n) \geq \delta)$*
- *Recurring Drift if $|H| \geq 3 \wedge \neg \text{IncrementalDrift} \wedge \exists m \in \mathbb{N}, 2 \leq m \leq n, |S_{M_n, T} - S_{M_{n-m}, T}| \leq \epsilon$*
- *Gradual Drift if $U \neq \emptyset$*
- *Sudden Drift if $\neg \text{Gradual Drift}$*

Following this definition, an incremental or recurring drift can either be a gradual or sudden drift as well, e.g., if there are still process instances using the previous process model, it is a gradual drift. Otherwise it is a sudden drift.

Algorithm 2 focuses on identifying the type of drift. As input parameters a list of traces T , the traces from the `trace_map`, a list of unfinished traces U for M_{n-1} , collected by Alg. 1, for detecting gradual drifts, a process history H_P , δ for determining fit traces and ϵ are required. ϵ describes the maximum error that is allowed between two model scores to be equally viable for T and ranges from 0 to 1, where 0 only determines equal scores to be similar viable and 1 determines any scores to be similar viable.

If there are less than two process models in the process history, it can be concluded that there is no concept drift, since a drift appears when the business process logic changes and a new model is discovered.

For every process model of H the model score is calculated using traces from T , like described in Alg. 1. The variable **Incremental** is calculated during the calculation of the scores to save execution time. If “Incremental” equals 1 an incremental drift is detected, otherwise not.

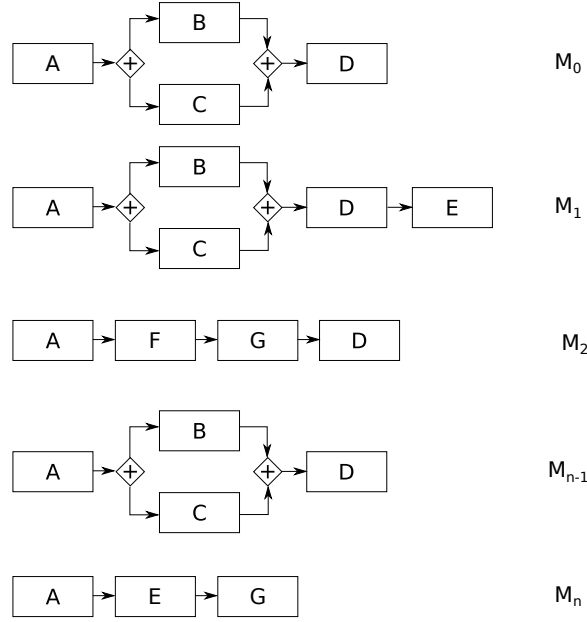


Figure 4.7: Complete process history of a single business process

If there are traces out of T that fit the preceding model and the current model, an incremental drift is detected. As long as U is not empty, the incremental drift is a gradual drift as well. Otherwise it is a sudden incremental drift.

For recurring drifts, the score of any model from M_{n-2} to M_0 is calculated. If there is at least one model M_m , where the difference between $S_{M_n,T}$ and $S_{M_m,T}$ is less or equal ϵ , a recurring drift is detected. Then it is again distinguished between a gradual recurring drift and a sudden recurring drift, using the same approach as before.

If it is not a recurring drift or an incremental drift, the number of elements in U is checked. If there is at least one trace, a gradual drift is detected. Otherwise it is not a gradual drift and a sudden drift is detected, since it is already concluded that it is not an incremental or recurring drift either.

The return value is a vector with 4 items corresponding to **Incremental Drift**, **Recurring Drift**, **Gradual Drift** and **Sudden Drift**. E.g., a gradual recurring drift return $[0,1,1,0]$, while a sudden drift returns $[0,0,0,1]$.

In Fig 4.7, a complete process history is shown with $\epsilon = 0.05$ and $\delta = 1$. The first concept drift from M_0 to M_1 is detected with T containing t_1 [A,B,C,D,E] and t_2 [A,C,B,D,E]. An incremental drift can be detected between M_0 and M_1 , since there is only a new event, E, added to the end of the process. The same traces that fit M_0 , fit M_1 as well.

Let $t_{2,3}$ [A,B,C,D], $t_{4,5}$ [A,C,B,D] and $t_{6,...,9}$ [A,F,G,D] be new traces in the event stream. With a small window size k , e.g. 5, M_2 is mined. Only $t_{6,...,9}$ are considered for the model, since they are not fitting M_1 . The difference between $S_{M_2,t_{5-9}}$ and $S_{M_1,t_{5-9}}$ or $S_{M_0,t_{5,...,9}}$ is greater than ϵ , so it is not a recurring drift. There are no traces fitting M_2 and M_1 as well, so an incremental drift is not possible. Since t_5 is likely to be not finished for M_1 , a gradual drift is detected.

Assume the next traces in the event stream are t_{10-12} [A,B,C,D] and t_{13} [A,C,B,D]. This results in M_{n-1} . The difference between $S_{M_0,t_{9,...,13}}$ and $S_{M_{n-1},t_{9,...,13}}$ is 0. A recurring drift is detected with the recurrence of M_0 . Since t_9 is already finished, it is not a gradual

drift as well.

Let the next traces be, $t_{14,\dots,17}$ [A,E,G], which result in M_n . There is no equally similar score to $S_{M_n,t_{13,\dots,17}}$ and there are no unfinished traces.

In the next section, Algorithms 1 and 2 are evaluated on a synthesized log, following the insurance example used in [BvdAŽP11].

4.1.3 Evaluation of process histories

For this evaluation, process execution log files have been synthesized, transformed into an event stream and the process history discovered. The business process describes an insurance process, first seen in [BvdAŽP11]. This evaluation focuses on the feasibility of process histories. The first part of this section covers the implementation of the algorithms and the framework for the evaluation. The second part shows the execution and results.

Implementation: To synthesize log files and an event stream, the algorithms from Chapter 4 are used. A web service is generated, which listens to the event stream. This web service, written in Ruby [MI02], processes each event and runs both algorithms described in Section 4.1.2. The process history is constantly adapted and provided through a REST interface. For the mining algorithm the inductive mining algorithm is used, implemented in the ProM extension RapidProm [vdABvZ17]. The model is then retrieved using the REST interface.

Evaluation: For the evaluation, process execution log files, based on the process models used in [BVDAZP14], have been synthesized. Small modifications have been applied, because only one path of some decisions showed concept drifts. For every process model 100 process instances were created. Since not all types of drifts are detectable in these models, new process instances to find every type of concept drift have been added. For the creation of the process history, k was set to 50, δ to 1 and σ to 0.9. For the distinction of a concept drift, ϵ was set to 0.05 and σ to 1.

The first 100 process instances consists of “Register”, “Decide High/Low”, , “High Insurance Check”, “High Med.History Check”, “Contact Hospital”, “Prepare Notification”, “By Phone”, “By Email”, “By Post” and “Notification Sent”. The order of “High Insurance Check”, “High Med. History Check” and the order and existence of “By Phone”, “By Email”, “By Post” have been randomised, so that the inductive miner is able to detect the parallel paths and decisions. The first models produced can vary a lot, depending on the order of events in the event stream. Fig. 4.8 shows the first discovered viable process models in the process history. M_0 consists of only one event. During the first 100 instances, the process model evolves and, depending on the order of the execution of the process instances, the first part of the first parallel gateway can be seen in M_4 . Algorithm 2 detects for the first process models in the history only incremental drifts, as expected. This can be reasoned because, every time a new event is found at the end of a trace or a new parallel order instead of sequence is mined, all other traces from the previous model are fitting the new model, e.g, the trace [“Register”, “Decide High/Low”, “High Insurance Check”, “Contact Hospital”] and the trace [“Register”, “Decide High/Low”, “Contact Hospital”, “High Insurance Check”] are both fitting M_5 .

After each possible combination is executed, M_{n-4} (Fig. 4.10), is discovered. To create a gradual drift, the next 50 instances are fitting M_{n-4} , but did not finish before the next 100 instances started in the stream, containing small adaptations. Instead of a parallel gateway for the medical checks, cheaper checks, like “High Insurance Check”, are done

Input: Traces *traces* (list of traces), Traces *u* (list of unfinished traces), *H* (Process History),
 δ (Threshold for fitting models $\in [0,1]$)
 ϵ (maximum error between similar process models)

Result: **type_vector[0,0,0,0]** (Positions represent Drifts
[Inc,Rec,Grad,Sudden] 1 represents this type of drift occurred.)

```

if H.size <= 1 then
  | return "Error: No drift"
end
Scores = [ ]
Incremental = 0
for M in H do
  | model_score = 0
  | for t in traces do
    | if conformance_checking(t, PM) >=  $\delta$  then
      | model_score += 1
      | if M == Mn-1 and conformance_checking(t, Mn) >=  $\delta$  then
        | Incremental = 1
      | end
    | end
  | end
  | Scores.append(model_score/traces.size)
end
Scores = Scores.reverse // Reverse order so Scores[0] == Mn
if (Incremental == 1) then
  | if u.size ≠ 0 then
    | return [1,0,1,0] //Incremental Gradual Drift
  | else
    | return [1,0,0,1] // Incremental Sudden Drift
  | end
end
for i in Scores.size do
  | //Start with 0 if i ≤ 1 then
    | next
  | end
  | if (|Scores[0]-Score| ≤  $\epsilon$ ) then
    | if u.size ≠ 0 then
      | return [0,1,1,0] //Recurring Gradual Drift
    | else
      | return [0,1,0,1] // Recurring Sudden Drift
    | end
  | end
end
if u.size ≠ 0 then
  | return [0,0,1,0] // Gradual Drift
end
return [0,0,0,1] // Sudden Drift

```

Algorithm Part 2: Algorithm for identifying the specific type of concept drift.

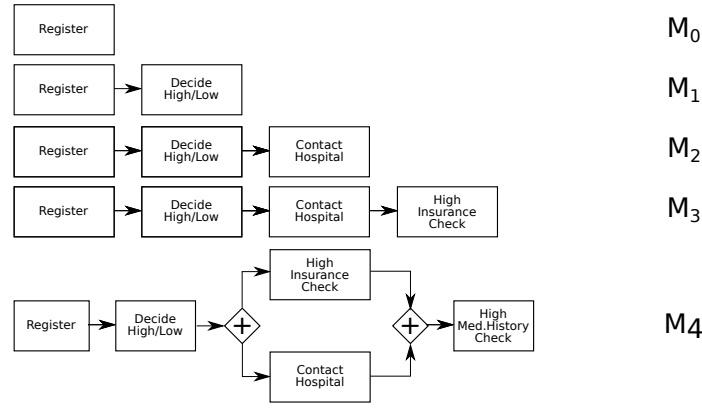


Figure 4.8: Process Models containing concept drifts.

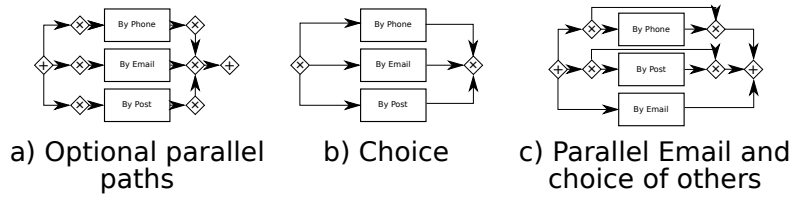


Figure 4.9: The concept drift from a to b is not detectable as well as the drift from a to c, since traces from b and c fit a. The drift from b to c is detectable.

at the beginning. If this check fails, the other checks are automatically skipped. Unfit traces now contain only a subset of the 3 events. After 45 instances, σ and the score of the new process model M_{n-3} for the traces of the trace_map are equal, so the model is appended to the process history. Since there are 5 instances for M_{n-4} not finished as well, a gradual drift is detected. An incremental drift has been detected as well, since some traces fit M_{n-4} and M_{n-3} perfectly.

All the concept drifts from [BvdAŽP11] cannot be detected with our approach. As can be seen in Fig. 4.9, the first process model includes the events “By Phone”, “By Email”, “By Post” in optional parallel paths. The first concept drift described in [BvdAŽP11] changes the parallel gateway to a decision, where only one event is chosen (Fig. 4.9(b)). Since all paths are optional anyway, all traces fit, even if only one event is present. To negate this, a periodical model could be mined, using all traces in the trace_map to detect a stricter model fitting all traces. The other model containing again a subset of choices already possible in the parallel optional model, suffers from the same problem. The concept drift from Fig. 4.9 (b) to (c) could be detected, but only if b is discovered. The drift from (a) to (c) cannot be detected.

Another 100 instances have been injected into the event stream, representing a new legislation. The split of high and low insurance claims has been removed, i.e., every claim is treated the same way. The notifications are only allowed to be sent per post as well. After 45 instances, the model M_{n-2} is discovered. This model varies vastly from M_{n-3} , since the score from M_{n-2} is 0.9 and the score of M_{n-3} is 0.1. No traces from M_{n-3} match M_{n-2} . Also M_{n-2} does not conform to any other known model in the process history. A sudden drift is detected.

In the next 100 instances, a new event at the end has been discovered. “Receive delivery

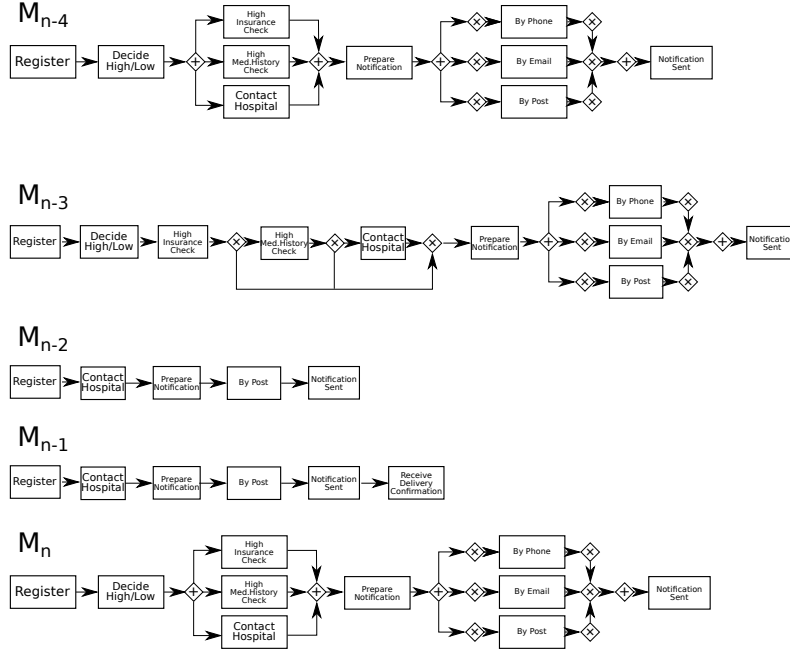


Figure 4.10: Process Models containing concept drifts.

confirmation” is appended to the end of the new process instances. Again after 45 instances, M_{n-1} is discovered. Since the 5 oldest traces still fit M_{n-2} and M_{n-1} an incremental drift is detected.

For the last 100 instances, the first 100 instances have been injected into the stream again with modified time stamps. As expected, after 45 instances, M_n is discovered, which is identical to M_{n-4} and both have the same score of 0.9. A recurring drift has been detected.

All four types of concept drifts can be detected. A problem occurs, if the process model after the concept drift is just a stricter model. This means if new traces fit the current model perfectly, no new model will be discovered and no concept drift will be detected. This can be negated by discovering a new model periodically instead of only if an unfit trace has been found, but this could lead to big mixed process models, if not only the unfit traces are used for discovering a new model. E.g., the sudden drift in Fig. 4.10 from M_{n-3} to M_{n-2} , could also be interpreted with a decision after the “Register” event, which leads to the path from M_{n-3} or the path from M_{n-2} .

4.1.4 Summary and Outlook of Process Histories

This section introduces process histories to reflect the evolution of a process based on an event stream during run-time. The histories consist of a sequence of viable models of this process. Based on this model sequence, incremental, sudden, recurring, and gradual concept drifts can be detected. For synthesizing the process histories, an algorithm utilizing conformance checking and the “age” of event information has been presented. The feasibility of all concepts are evaluated through a proof-of-concept implementation. With static log files [BvdAŽP11], the exact point of time of a concept drift can be detected, but is not using an online environment and does not differentiate the types of concept drifts. In an online environment [MBCS13] [vZBH⁺17], concept drifts can be detected, but not all

types of concept drifts have been covered and the drift is detected relatively late. The advantage of the other approaches is the detection of stricter process models, since they are not focused on detecting drifts, but discovering new process models. Future work will focus on the refinement of synthesizing process histories, i.e., parallel events at the end of a process, other techniques to calculate the fitness of a specific trace, detecting concept drifts in a stricter model as well as testing other mining algorithms including the frequency of events and other approaches like lossy counting instead of sliding window for the determination of impactful traces. The next section focuses on process histories which take the data elements of events into account.

4.2 Concept Drifts on the Data Perspective

Flexibility and change are still among the most pressing challenges for processes [RW12]. This holds particularly true for data-driven process executions in volatile environments such as manufacturing processes [PMRP18]. Manufacturing processes control and are controlled by a multitude of data, e.g., machining parameters and sensor data for example, that constantly monitor the state of the process and the machines. Changes in these parameters are common due to, for example, environmental changes or errors, and can be of tremendous importance for the quality of the process and the product. Similar requirements hold for patient treatments where shifts in vital parameters have to be detected immediately. Hence it is of great importance to be able to detect changes in the data attributes of processes, specifically during run-time, i.e., based on process event streams.

This necessitates making a next step in detecting and evaluating so called concept drifts [BVDAZP14]. So far concept drift refers to changes in the control flow of the process that are discovered based on process execution logs. In the previous section, algorithms for detecting and representing concept drifts in control flow from *event streams* are provided. This section aims at detecting changes in process data, called data drift in the following, from process event streams at run-time. This is necessary as detecting data drifts from process execution logs ex-post might be too late in order to take necessary actions in many cases.

Generally, data drifts can be categorized following the same guidelines gathered from [BVDAZP14]: data drifts can have recurring effects as well as incremental effects or just reflect sudden changes in the business process logic.

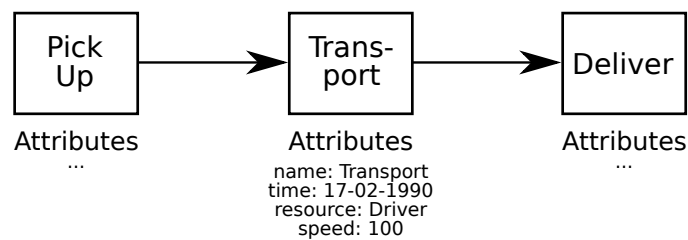


Figure 4.11: Process model with data attributes of event **Transportation**

Figure 4.2 shows a process example from the logistics domain. A product is picked up by a delivery service, transported and delivered to the customer. The data attributes for

the event **transportation** are timestamp, name of the event, resource that is executing this event, and average speed. Assume that the system is detecting that a significantly decreased speed is detected in events of new process instances. The reason for this can be manifold, like a construction site on the road, or even a construction site on a different road, which causes the normal route to be jammed. The control flow of this process is not changed, but the data attributes show a drift in the execution of the process, a data drift. Detecting such drifts early helps tremendously in finding errors and bottlenecks that suddenly occur. A data drift could also reflect the natural evolution of a process, e.g., instead of only doctors, nurses administer drugs as well, due to a legislation change. This would be reflected in a new organisational role for this event.

Similarly to control flow drifts (cf. Section 4.1), data drifts can have different effects, i.e., recurring as well as incremental effects or they just reflect sudden changes in the business process logic. Moreover, data drifts must be detected during run-time and not ex post for many application domains where immediate action is required. Finally, data-intense processes are often emitting a huge amount of events in high frequency.

Note that the problem is two-fold: First, a data drift needs to be detected. Afterwards, the of the data drift, e.g., recurring needs to be identified. For addressing the first part, the already established concept of **process histories** (cf. Algorithm 1 and 2, is extended to store information on process data attributes and to allow the detection of data drifts. These drifts are identified using outlier detection on the values of a data attribute. The approach can independent of the control flow of the process if instead of a model, only event attribute pairs are saved. This would yield the disadvantage of not seeing the data drifts as the evolution of a process without the process history. For the latter part, a formal definition for the data drift types is provided and an algorithm that determines the type of a data drift based on process histories. Therefore two algorithms are presented in this section. One of them synthesizes the extended process history in order to detect the data drift and the other one determines its type. They are evaluated through a prototypical implementation and application to a real-world data set from the manufacturing domain.

Section 4.2.1 provides fundamentals. In Sect. 4.2.2 the definition of data drift types and two new algorithms are presented. This section is followed by the evaluation in Sect. 4.2.3. An outlook and summary is provided in Section 4.2.4.

4.2.1 Fundamentals for Detecting Data Drifts

Process histories (cf. Section 4.1), are extended to comprehend viable data attributes into the process history and to detect new types of drift, data drifts.

The main contribution of this section focuses on events and their data attributes. Common attributes would be the point of time when an event has been executed, a organizational resource that has executed the event, or other arbitrary data attributes, e.g., the cost of an activity.

So far, only control flow drifts are captured in a process history, in fact data attributes are rarely considered except some exceptions like the decision mining algorithm[RvdA06]. In order to enable the detection of data drifts, **process histories** can be extended as follows.

Definition 4 (Data-extended Process History). *Let P be a process and ES the corresponding process event stream. A process history*

$H_P := \langle M_0, M_1, \dots, M_{n-1}, M_n, \dots \rangle$ is a list of viable process models $M_n, n \in \mathbb{N}$ that have been discovered for P from ES with M_n being the current model for P . $M \in H_P$ is defined as

$$M := \langle E, \langle (e_0, A_0^c), \dots, (e_k, A_k^c) \rangle \rangle, e_i \in E \text{ with}$$

- $E \subseteq ES$ is the set of all events in M ;
- $e_i \in E : e_i = (l_{e_i}, A_i)$, i.e., an event stores its label l_{e_i} and the set of data attributes A_i ;
- For $e_i, A_i^c \subseteq A_i$ denotes the sub set of attributes from A_i that have caused the data drift.

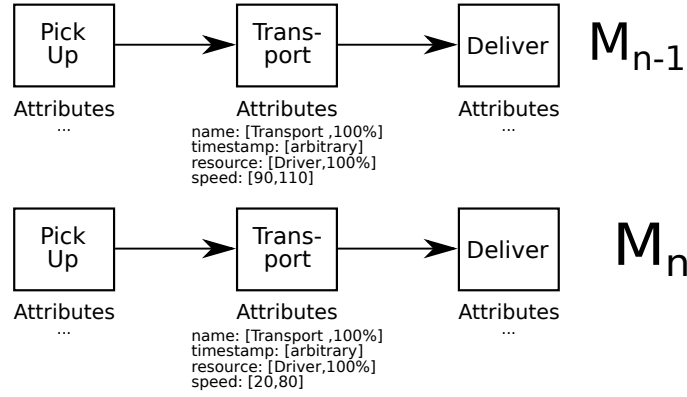


Figure 4.12: Process History showing a data drift in the attribute **speed**.

Figure 4.12 shows the extended process history for the example of Figure 4.2. The control flow of the models M_n and M_{n-1} is not changed, but still a new model has been detected because of a data drift in the event **Transport**. As can be seen, the lower bound for the average speed in M_{n-1} equals 90 and the upper bound equals 110. A number of outliers have been detected, e.g., 40, 40, 40, 40, 50, 50, 50, 60, 60, 60, 60, 50, 50, and 50. This results in the new lower bound 20 and the upper bound 80. The data extension does not interrupt the detection of control flow drifts as presented in Section 4.1.

For the algorithm the data structure **trace_map** is used. In addition, the data attributes are now stored as well for each event in the **trace_map**. In the following algorithms, this map is synthesized using an event stream. To detect the currently relevant traces in an event stream, the sliding window approach is used. This means that only k traces are considered for the detection of drifts. If a new trace is detected and there are already k traces in the **trace_map**, the oldest trace is removed and the new trace is stored.

Concept/Data Drifts: The 4 different types of concept drifts at the control flow level can also be defined and detected at the data level, which is explained in detail in Sect. 4.2.2. Concept drifts on the data level are called data drifts in this thesis.

4.2.2 Detecting and Identifying Data Drifts

In this section, the synthesis of data-extended process histories as basis for detecting and identifying data drifts is elaborated.

Detecting Data Drifts

Assume a process history $H_P = \langle M_0, \dots, M_n \rangle$ as defined in Def. 4 with most current process model M_n and the corresponding `trace_map`. The difference between M_n and M_{n+1} yields the data drift and its type. As basis, for each new event in the stream, the data attributes are checked for changes. In this work, changes in data attributes are detected based on outlier detection in the data attribute values. For this statistical methods will be used. However, it is not feasible to compare every new event to all previous events in all traces as this might be too complex and might lead to misleading results in terms of the drifts. Imagine that a change happened in one event and later the inverse change occurs. Considering all traces this change would not be detected as a drift. Hence, it is feasible to restrict the set of considered events and traces. In Section 4.1, the idea of using a sliding window on the traces has been proven promising and hence this concept will be applied for the synthesis of process histories in the following as well.

Algorithm 3 implements the core ideas of using a sliding window on the traces and outlier detection on the data attributes. As input an event stream, ES , a window limit k and the thresholds ϕ and κ are required. The thresholds are described in detail in the following paragraphs. The algorithm is used while synthesizing a process history. A data drift is detected after the detection of a control flow drift; algorithms for detecting control flow drifts are provided in Section 4.1. At the beginning of Alg. 3, process history H_P is an empty list and does not contain any process models. Also the `trace_map` which is used in the detection of data drifts does not contain any items in the beginning.

The sliding window technique, allows to identify currently significant traces for the detection of new viable models, where k is the maximal number of traces stored in the `trace_map`. The data extension uses the same window for detecting drifts in the data elements. Since outliers shall be detected, a certain amount of values for a specific data element, respectively a certain number of an events, needs to be detected for statistical analysis. The minimum number of events, κ , is user defined and a value between 0 and k , since it is, except for a loop, impossible to have more events stored, than there are traces in the `trace_map`.

After the event has been stored in the `trace_map`, the algorithm tries to detect a data drift. A whole new range of drift types is possible if a concept drift and a data drift occur simultaneously, which require a definition and an algorithm to be detected. This approach is planned to be researched in future work.

If the process history contains at least one model, a copy of the current model and its events with attributes is created. At the start the `list_of_data_drifts` is an empty list and contains pairs of the drifting attribute and its corresponding event. If the current model of the process history contains the currently processed event, an iteration over the data attributes of this event starts. In this iteration, a denotes a data attribute of currently processed event e . The next expression checks, if a is an outlier to e of the current model.

For the outlier detection following methods are used. If the data attribute a contains continuous data, the data could be transformed into a normal distribution [CG01] and a range is calculated. Outliers are defined for this approach at 1.5 times of the interquartile range below the first quartile and 1.5 times of the interquartile range above the third quartile [HV08]. The implementation currently only supports continuous data. If the data attribute a does not contain continuous data, we use the likelihood. If for example, only 3 equally common values have been detected in the last model for this attribute in 50 events,

and a new value occurs, its likelihood is lower than all of the known values. On the other hand, if there are 50 different values for one attribute in 50 events, it could be deduced that this attribute is arbitrary. A user input, defining the maximum distance between the new likelihood and the average likelihood of choices, is used as threshold, to detect outliers for this. If this attribute is not in the last known model for event e , the outlier function automatically returns true.

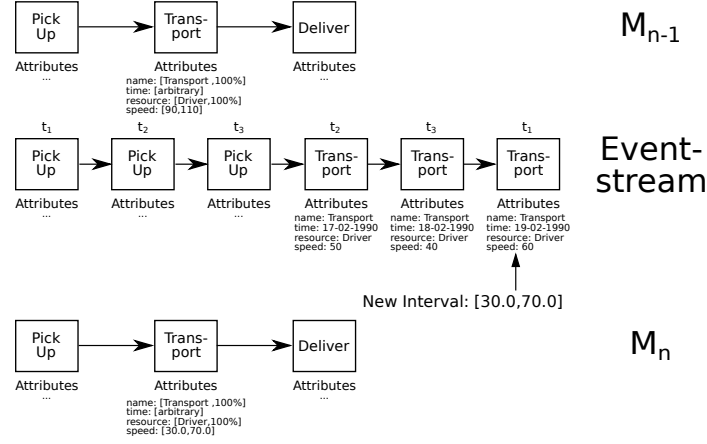


Figure 4.13: Synthesising a process history with $\kappa = 1$ and $\phi = 1$

In the next step an empty list `list_a` is created and the variable `as` is initialised with 0. This variable counts how often event e is found in the `trace_map` containing a . The algorithm searches every trace in the `trace_map`. If an event is found that equals e and also has the same attribute a as an outlier, this attribute is added to the list.

If the number of occurrences for attribute a in the `trace_map` (`as`) is smaller than κ , a data drift has been detected. Apparently this data attribute is not used often enough to retrieve significant information and is removed from the new model. The pair e, a is appended to the `list_of_data_drifts`

Otherwise, the new range or likelihoods will be calculated using only the information of outlying attribute values. It is then counted how often an attribute of the `trace_map` fits the new properties and is divided by the number of events e . This yields a score value, which represents the percentage of fitting attributes for the new properties. If this score is greater or equal than ϕ , the new properties are added to the new model and the pair e, a is appended to the `list_of_data_drifts`. The threshold ϕ is in $[0, 1]$, where 0 would be everything and 1 would be only considering scores, where 100% of the attributes match the new properties as a data drift. Afterwards, Algorithm 4 is executed, to detect the type of the data drift.

Figure 4.13, shows how an outlier is detected for the running example Fig. 4.2 and how and when a new model is appended. The range from 90 to 110 has been detected earlier. In the event stream three new traces are occurring, each of them having an outlier in the event **Transport**. With a sliding window size of 3, only outliers are in consideration for new models. Each time an outlier is detected, a new range is calculated if there are more or equal κ outlier in the sliding window. When the third outlier is detected, this requirement is met and the new range from 30 to 70 is calculated. Each of the currently viewed speed values are fitting this range. A new model is appended to the process history.

Data Drift Identification

Algorithm 3 detects data drifts in an event stream and creates new models for the process history. Every time a new process model is appended to the process history a data drift is detected. The four types of data drifts, in relation to a process history, can be defined formally as follows:

Definition 5 (Data Drift Types). *Let U be a given set of unfinished traces. Moreover let H be a process history for a process P containing only data drifts, which can be easily filtered by checking if the list of data drifts in a Model M is $\neq \emptyset$. Let $H_{dd} \subseteq H$ be the models of the process history containing data drifts and for $M_n = \langle E, \langle (e_0, A_0^c), \dots, (e_k, A_k^c) \rangle \rangle \in H_{dd}$ let $M_n.drifts := \langle (e_0, A_0^c), \dots, (e_k, A_k^c) \rangle$ yield the list of event attribute pairs, containing the attributes which have shown the data drift. Let $\phi \in [0, 1]$, $\sigma \in [0, 1]$ be thresholds, the function *outlier*, defined for a model and a data attribute, yielding true or false and the function *similarity*, defined for two attributes of an event, ranging from 0 to 1. The following drift types are defined as follows:*

- *Incremental Drift if $|H| \geq 2 \wedge \exists (e, A) \in M_n.drifts, (A \not\subset M_{n-1}[e] \wedge A \subset M_n[e]) \vee (A \not\subset M_n[e] \wedge A \subset M_{n-1})$*
- *Recurring Drift if $|H| \geq 3 \wedge \exists m \in \mathbb{N}, 2 \leq m \leq n, M_{n-m} \forall (\{e, A\} \in M_n.drifts, similarity(M_n[e].A, M_{n-m}[e].A) \geq \sigma$*
- *Gradual Drift if $|H| \geq 2 \wedge \exists t \in U, \{e, A\} \in M_n.drifts, \neg outlier(M_{n-1}, t[e].A)$*
- *Sudden Drift if $\neg GradualDrift$*

As a fitness function the same technique as in Section 4.1 using conformance checking with only considering moves in the log [VdAAvD12] is used. The similarity function checks if the statistical properties are alike. For example, if the intervals have a tremendous overlap or the distribution of likelihoods is similar.

It should be noted, that in this definition of data drift types, only the sudden and the gradual drift are distinct. It is possible for a data drift to be an incremental drift and recurring drift at the same time, e.g., a new data attribute has been detected in comparison to the last model, but this data attribute is also available and similar to an even older model. In the following, Alg. 4, is explained in detail and shows how to answer RQ3.

As input parameters a list of unfinished traces U for M_0, M_{n-1} , a process history H_P and σ are required. σ describes the threshold for determining if two statistical properties are alike and ranges from 0 to 1, where 0 determines any 2 properties as equal and 1 determines only exactly equal properties to be similar.

If there is only one process model in the new process history no data drift had happened. The first distinction is made between a gradual drift and a sudden drift. It has to be either of them, so if it is a gradual drift, it cannot be a sudden drift and vice versa. For this, the algorithm iterates over the `list_of_data_drifts` of the current model. If there is a trace out of U for which its attributes and events match an entry in the list and is not an outlier, if compared to the second to last model, M_{n-1} , a gradual drift is detected, because there are still unfinished traces, that corresponds to the older model. The outlier function is the same, like in Alg. 3. The third position in the return vector is set to 1,

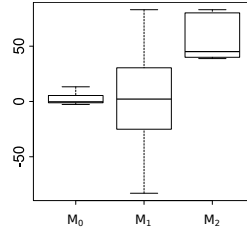


Figure 4.14: Results reflecting the range of the torque value

which signals a detected gradual drift. Likewise it can be determined if it is not a gradual drift, a sudden drift is detected.

In the next step, the `list_of_drift_events` is again iterated. If an attribute is not found in the older model M_{n-1} or if an attribute is not found in the current model M_n , it can be deduced that the attribute has been added or removed respectively. This indicates an incremental drift, represented by a change the value of the first element to 1 of the return vector.

If there are at least 3 process models in the history, a recurring drift can be detected. If there is at least one model from M_0 to M_{n-2} where all attributes of the `list_of_data_drifts` are similar, a recurring drift is detected. The resulting vector is returned at the end containing the information on which data drift could be detected. In the next section, the two algorithms are evaluated on a real life log, using a log from the manufacturing domain.

4.2.3 Evaluation of Data Drift Detection Algorithms

For evaluating the approach, a prototypical implementation in Ruby [MI02] is used and applied on a real world process execution log from the manufacturing domain. The underlying process executes the manufacturing of small metal parts for different machines.

Algorithms 3 and 4, are integrated into the algorithms presented in Section 4.1. The steps of creating the `trace_map` and using a sliding window have been merged from Algorithm 1 into Algorithm3 and Algorithm 4 to save computation time.

The log files from the real world example are stored in XES format and consist of 10 process instances containing 40436 events in total, but instead of being serialized in XML, the log files are serialized in YAML [BKEI05]. The process execution log has been transformed into an event stream. The models in the process history have been discovered, using the approach in Section 4.1.

For this evaluation the event “AXIS/Z/aaTorque” is chosen and the data attribute “value” is looked at. This event appears 4415 times in the log files in total and is numeric. The only available non-numeric data attributes in this log file, reflect either an enumeration, where only specific values are allowed or an arbitrary value.

The event “AXIS/Z/aaTorque” describes the positioning of the machine part in the z axis. With the sliding window k set to 200 and κ to 100, the first boxplot, seen in Fig. 4.14 (M_0), has been detected. For the outlier detection, the length of the whiskers, the interval $[-11.05, 15.28]$ has been calculated. Using 0.9 for the threshold ϕ , 2 data drifts have been detected, at the 123rd and 3187th time the event appeared in the event stream, shifting the boxplot to Fig. 4.14 (M_1) and (M_2) with the new intervals $[-105.10, 38.75]$ and $[-20.28, 89.99]$, two significant changes in the business process logic. This could be

caused by a different part being produced in the machine using different values, or the replacement of a part of the machine where the new part is using new parameters.

Using a less strict drift detection threshold ϕ with 0.8, 8 drifts have been detected. The ranges of the intervals differ greatly, where only the fourth drift, when the event appeared for the 1795th time and the last drift, when the event appeared for the 2800th time could be suggested as a recurring drift. The intervals $[-105.10, 38.75]$ and $[-117.42, 91.08]$ overlap about 68%. Since there is always the same number of data attributes in the event, caused by the process execution engine which saved these logs, the only incremental drift is always the first one at the κ th time the event occurs, since in the previous model the data attribute is absent. All of these drifts are gradual drifts, because the drift never occurred in the last appearance of an event of a process instance.

This evaluation was carried out with a proof of concept implementation to present data drifts in a data attribute of an event and the determination of its type. This procedure can be reproduced with any number of attributes of events, yielding a new model with adjusted statistical properties for the drifting attributes.

4.2.4 Conclusion for Data Drift Detection

An extension to process histories to include data attributes and to detect and identify data drifts from event streams has been introduced. Data drifts are part of the evolution of business process, therefore a data drift can be categorized into the four categories of concept drifts, i.e., incremental, recurring, gradual, and sudden. All four types can be detected and are formally defined. Two new algorithms have been presented. The first one synthesizes a process history with data attributes. The other one allows to determine the type of data drift. The evaluation shows promising results. Based on a prototypical implementation and a real-world data set from the manufacturing domain it is possible to detect data drifts. The future work includes a more user friendly implementation of the algorithms and testing the algorithms on more data sets.

4.3 Conclusion and Outlook

This chapter investigates the changes a process model experiences in its lifetime, on the control flow and on the data level. To summarize, the main contributions of this chapter are:

- In Section 4.1, two algorithms are presented to identify concept drifts and determine the type of concept drift occurring. Additionally the concept of a process history is presented, e.g., a collection of all the process models and concept drifts. The concept is evaluated on a synthetic log.
- Section 4.2 enriches the previously defined concept of a process history by defining data drifts, i.e., concept drifts on the data level instead on the control flow level. In this section, two algorithms are provided to identify and categorize data drifts and are evaluated on a real-life process log.

Each process model in a process history is the result of a drift. This could be a concept drift, a data drift or even both at the same time. Future work is planned to use process

histories to predict changes in a process, i.e., predict a recurring drift happening, e.g., a change to an older process model of the process. In Chapter 5, conformance checking algorithms are being investigated and extended by external data elements like data sensor streams as well as a more refined cost function for the conformance checking algorithms.

Input: Event Stream ES (a series of events)
 k (Limit for number of trace_map items)
 κ (Threshold for number of an attribute for consideration, $[0,k]$)
 ϕ (Threshold for distinction of a new viable data range $[0,1]$)

Result: Process History H_P (contains all viable process models in chronological order.)

$H_P = []$, trace_map < trace_id, trace > = 0

```

for e in ES do
  if trace_map contains_key e.trace_id then
    | trace_map[e.trace_id].append(e)
  else
    if trace_map.size ≥ k then
      | trace_map.delete_oldest
    end
    trace_map.insert(e.trace_id,e)
  end
  detect_concept_drifts_based_on_workflow_drifts();
  if |H_P| ≠ 0 then
    New_Model = H_P.last, list_of_data_drifts = []
    if H_P.last.contains(e) then
      for a in e do
        if outlier(H_P.last[e],a) then
          list_a = [], as = 0
          for t in trace_map.values do
            for ev in t do
              if ev == e and ev contains a then
                | as+=1;
              end
              if ev == e and outlier(H_P.last[e],ev.a) then
                | list_a.append(ev.a);
              end
            end
          end
          if as < κ then
            if New_Model[e] contains a then
              | New_Model[e].remove(a);
              list_of_data_drifts.append({e,a});
            end
            break;
          else
            e_size = 0; fitting_e = 0; properties = calc_properties(list_a);
            for t in trace_map.values do
              for ev in t do
                if ev==e then
                  e_size+=1;
                  if !outlier(properties,ev.a) then
                    | fitting_e+=1;
                  end
                end
              end
            end
            score = fitting_e / e_size;
            if score ≥ ϕ then
              | New_Model[e].a.properties = properties;
              list_of_data_drifts.append({e,a});
            end
          end
        end
      end
    end
    if |list_of_data_drifts| > 0 then
      | H_P.append(New_Model)
    end
  end
end
end
end

```

Algorithm 3: Algorithm to synthesise a process history

```

Input: Traces  $u$  (list of unfinished traces),  $H$  (Process History),
 $\epsilon$  (maximum error between similar statistical properties.)
Result: type_vector[0,0,0,0] (Positions represent Drifts [Inc,Rec,Grad,Sudden], 1 represents
this type of drift occurred. )
res = [0,0,0,0];
if  $|H| \leq 1$  then
|   return "Error: No drift"
end
//  $M$  is an abstraction to directly access the models of  $H$ 
for  $e, a$  in  $M.list\_of\_data\_drifts$  do
|   for  $t$  in  $U$  do
|   |   if  $!outlier(M_{n-1}, t[e].a)$  then
|   |   |   res[2] = 1 // Gradual Drift
|   |   |   break;
|   |   end
|   end
end
if res[2]  $\neq 1$  then
|   res[3] = 1; // Sudden Drift
end
for  $e, a$  in  $M_n.list\_of\_drift\_events$  do
|   if  $(!(M_{n-1}[e].contains\ a))$  then
|   |   res[0] = 1; // Incremental Drift
|   end
|   if  $(!(M_n[e].contains\ a))$  then
|   |   res[0] = 1; // Incremental Drift
|   end
end
if  $|H| \geq 3$  then
|   for  $m$  in  $(M_0, M_{n-2})$  do
|   |   bool found = false;
|   |   for  $e, a$  in  $M_n.list\_of\_drift\_events$  do
|   |   |   if  $similarity(\bar{M}_n[e].a, m[e].a) < \epsilon$  then
|   |   |   |   found = true;
|   |   |   |   break;
|   |   |   end
|   |   end
|   |   if found then
|   |   |   res[1] = 1; // Recurring drift;
|   |   end
|   end
end
return res;

```

Algorithm 4: Algorithm to identify data drift.

5 Time & Data-Aware Conformance Checking and Explaining Drifts

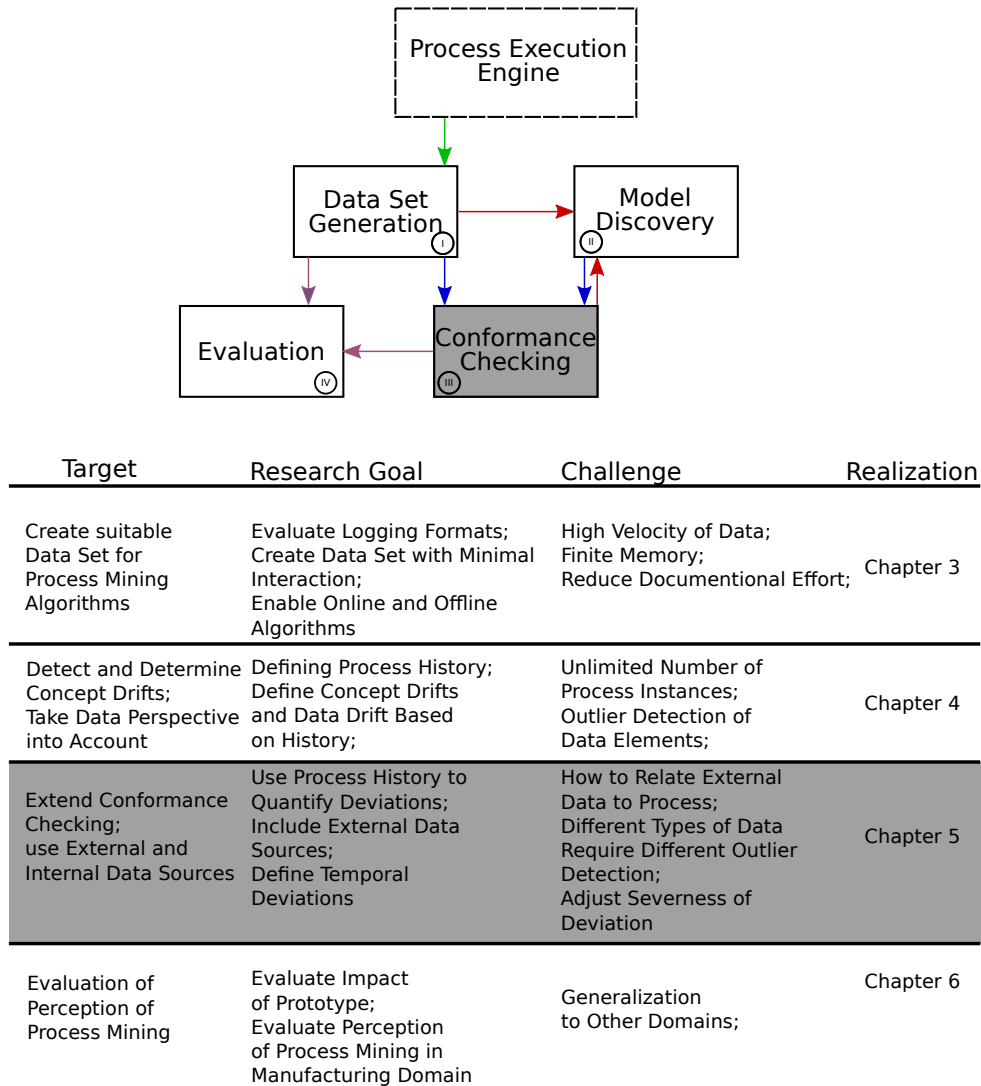


Figure 5.1: Overview of Targets, Research Goals, Problems of this thesis. The features for this chapter are marked.

This chapter addresses objective (III) in Figure 5.1, i.e., novel approaches for taking different perspectives and additional data into account for conformance checking.

The previous chapter, introduced algorithms to discover and identify drifts in a process using standard conformance checking algorithms and outlier detection to detect such drifts.

Figure 5.2 gives a brief overview of the three different perspectives of conformance checking this chapter is focusing on.

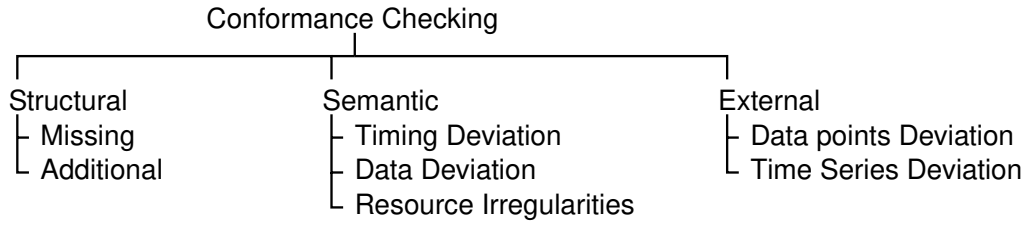


Figure 5.2: An overview of the three different areas of this chapter. Conformance checking on a structural perspective is targeting the order of event. Semantic conformance checking is focusing on other perspectives, i.e., the temporal perspective. The last area, aims at taken external data into account to discover the source of a drift in a process model.

This chapter investigates all three different branches depicted in Figure 5.2. A simplified version of the process model present in Figure 1.1 can be seen in Figure 5.3. Traditional conformance checking focuses on the structural branch of Figure 5.2, i.e., the order of events that are present in a process instance and assigns a cost for missing and additional events present in a process instance [CvDSW18]. Usually the cost for a deviation is taken any information into account, to adjust the severity of mismatch in a process instance, i.e., it can be argued that if a cholecystectomy is performed twice on the same patient, it is a more severe situation, than if the event for the hospital admission is missing. Yet, a mismatch can also occur at the attached data elements of an event and the point of time events are registered, i.e., the duration for an event or the time duration between events. The semantic branch of Figure 5.2, focuses on detecting deviations on these different perspectives. While there is already algorithms focusing on data deviations [MdLRvdA16a], temporal deviations are still untouched. Lastly, data outside the process perspective can influence the behavior of a process instance. The external branch of Figure 5.2 focuses on external data [DvC⁺16], that can be put into relation to a process instance, i.e., the blood pressure and heart rate measurements of a patient in a hospital are not related to a specific event in the process, but rather measured during the complete stay of a patient (cf. 5.3). This external data can contain information on the root cause of a concept drift, i.e., a canceled or deviating flight due to bad weather conditions [DvC⁺16].

The following three research questions are targeting all different areas of conformance checking seen in Figure 5.2:

- RQ ③a How to better quantify the costs of deviations between process instances and a process model using data elements?
- RQ ③b How can the temporal perspective be taken into account for conformance checking?
- RQ ③c How to discover the source of a concept drift?

Section 5.1 focuses on RQ ③a, an advanced approach to adjust costs of mismatches in conformance checking algorithms, reflected in the structural branch of Figure 5.2.

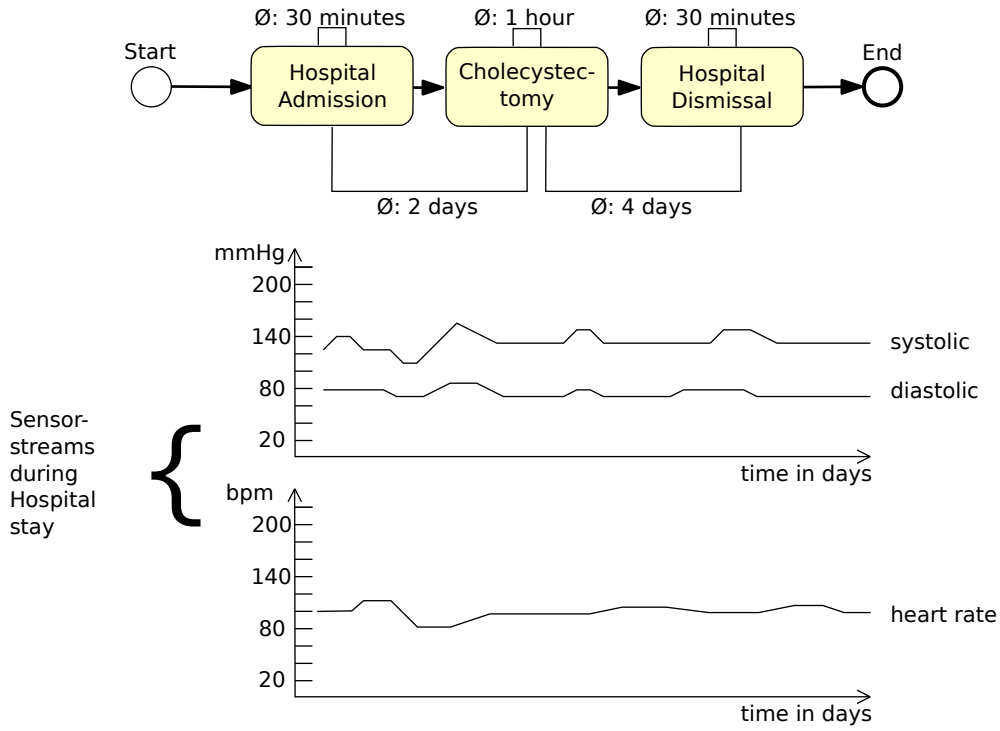


Figure 5.3: A smaller version of the running example depicted in Figure 1.1. Here only the tasks for patients with severe symptoms are used for demonstration.

Conformance checking algorithms analyze the behavior of a process instance compared to a process model and for every mismatch a certain cost is assigned. Currently these costs are assigned by a basic cost function. Section 5.1 presents a novel approach, to adjust the costs of mismatches using an advanced cost function. The cost is adjusted, based on the data elements attached to events, i.e., if the data elements do not contain any anomalies, a mismatch can be reduced to logging error instead of an execution error. The approach is evaluated on a real-world data set.

Section 5.2 concerns oneself with RQ (3b), the temporal perspective of a process for conformance checking, positioned in the semantic branch of Figure 5.2. It establishes new algorithms to detect anomalies in the temporal distances between different events in a process instance as well, as the execution duration of an event, e.g., a suspiciously small or huge temporal distance between two events or an extraordinarily short or long duration of an event. The approach is evaluated on a real-world data set.

Section 5.3 tackles RQ (3c) and focuses therefore on the external branch of Figure 5.2. As mentioned before, additional data elements outside the process can affect a process, e.g., the blood pressure and heart rate of a patient during a hospital stay. These external data sources can be interpreted as a stream of data points. Drifts in these streams, can be identified as the source for a drift in the process. Section 5.3 provides methods to include these external data streams into a process instance and presents a novel approach using dynamic time warping [RCM⁺12] to detect drifts in the data streams and to explain drifts in a process model. The approach is evaluated on a real-world data set.

A selection of text, figures and tables within this chapter is based on the following publications.

Stertz, F., Mangler J., Rinderle-Ma S.: Analyzing Process Concept Drifts Based on Sensor Event Streams During Runtime In: 18th Business Process Management, BPM 2020), Pages: 202–219
https://doi.org/10.1007/978-3-030-58666-9_12

Stertz, F., Mangler J., Rinderle-Ma S.: Data-driven Improvement of Online Conformance Checking. In: International Enterprise Distributed Object Computing Conference, EDOC 2020), Pages: 187-196
<https://doi.org/10.1109/EDOC49727.2020.00031>

Stertz, F., Mangler J., Rinderle-Ma S.: Temporal Conformance Checking at Runtime based on Time-infused Process Models. Tech. rep. (2020), <https://arxiv.org/abs/2008.07262>

5.1 Extending Conformance Checking Algorithms Using an Advanced Cost Function

As mentioned in the introduction of this chapter, conformance checking can affect different areas. This section focuses on the structural area, as depicted in Fig. 5.4. Structural conformance checking deals with missing/additional events or data. *Conformance checking* is designed to determine if the logged process execution of a business process correctly represents the desired execution specified by a process model [CvDSW18]. A conformance of 1 means that the behavior described by the model is perfectly reflected in the log and vice versa. A conformance < 1 indicates deviations in the behavior described by the model and reflected by the log. Then a conformance deviation has occurred. It is crucial to detect conformance deviation and to investigate their reasons in detail in order to, for example, be able to distinguish intended deviations and undesired ones. The latter might hint to security breaches or ad-hoc changes in process instances due to problems during process execution.

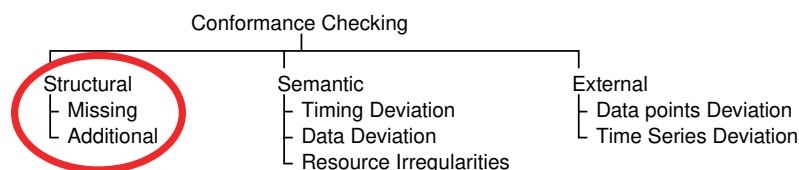


Figure 5.4: Conformance Checking Types and Affected Artifacts. This section focuses on the circled area.

Figure 5.5 shows reasons and causes which might lead to conformance deviations. An automatic differentiation between those causes as well as the quantification of individual deviations is the main motivation for this section.

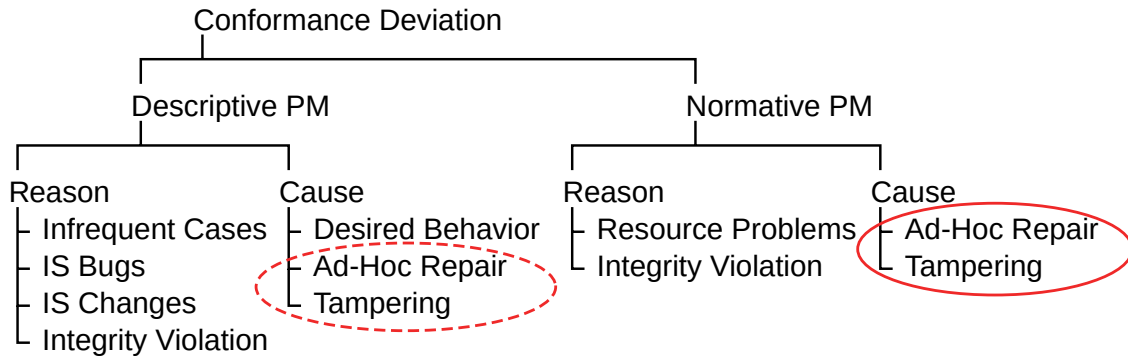


Figure 5.5: Scenarios for Conformance Deviations (PM: Process Model)

Process models can be descriptive, i.e., described by a rule set consisting of rules defining the order of tasks for example, allowing execution of more infrequent cases, since the rule set typically is not covering every option possible. On the other hand, process models can be normative, where the order of the events is strict and each execution sequence not following the process model is not correct.

In a scenario with a **descriptive process model**, Information Systems (IS) with hard coded process logic contribute to an event log. These IS may contain logic for infrequent cases, that are not yet fully described by the process model. These IS may furthermore contain bugs (erroneous code), or they may be still under development introducing new behaviors. Lastly the underlying IS may be compromised, leading to security violations, or tampering. With conformance checking it is possible to point out deviations which may then be attributed to one of the aforementioned reasons.

A scenario with a **normative process model** on the other hand is much less fuzzy. It can be assumed that the process model is actively enacted by some kind of process engine, thus most of the time the log will always be perfectly aligned with the PM. It can be furthermore assumed that PM changes only occur in line with conformance checking changes. Thus the causes for conformance deviations can typically be attributed to either problems with resources or security violations (tampering).

While previous chapters of this thesis concentrated on semantic aspects, this section will concentrate strictly on missing/additional events (see ○ in Fig. 5.4). In particular, an approach to quantify and classify conformance deviations is presented, as well as a method to automatically classify conformance deviations.

Thus, this section proposes a novel approach to quantify the overall conformance deviation cost of a single process execution for a given PM and a given log, based on per-deviation cost values assigned to missing / additional events occurring in a process log. These per-deviation cost values can then be automatically adjusted based on causes shown in Fig. 5.5. While *ad-hoc repair* typically is conducted in a way that it does not affect any subsequent events, *tampering* can lead to observable effects in the log. Thus, based on a set of well-defined effects observable in the process log, the cost values may be either decreased or increased. The per-deviation costs contribute to advanced cost function which yields a conformance deviation cost value. The conformance deviation cost value can then be utilized for automatic classification of causes as depicted through ○ in Fig. 5.5. In addition, this approach can be applied on an event stream as well, instead of

Definition 6.

$$advanced_cost_function(x) = \begin{cases} 0, & \text{if no deviation} \\ b_m - \sum_{n=0}^{len(de)} (\omega_n * b_m * \chi_{D_n}(de_n)), & \text{otherwise} \end{cases}$$

Let b_m be the base cost per deviation and de , a set of selected data_elements attached to an event. D is a set of acceptable values for the data elements for this event, ω any number between 0 and 1, and χ the indicator function. This function returns the altered cost for a preceding deviation.

a process log, to detect tampering as soon as possible, therefore help the user repair the process more efficiently.

While the proposed approach is valid for both, descriptive and normative PMs, the evaluation has been conducted based on a normative PM and a process log produced by a process engine. For the evaluation an artificial dataset, as well as a real-world dataset from the manufacturing domain have been used.

Section 5.1.1 introduces the advanced cost function and a possible implementation of a algorithm incorporating it. The contribution is then evaluated and discussed in Section 5.1.2.

5.1.1 Advanced Cost Function

In this section the main contribution is explained in detail, i.e., the definition of the advanced cost function and how to gather the relevant data out of a process execution log.

Advanced Cost Function

Conformance checking aims to align a sequence of events of a process log to a possible event sequence of a process model. For every deviation, i.e., an unexpected event in a sequence, a cost for the deviation is assigned. The alignment cost of a event sequence of a process log consists of the sum of all deviation costs. Currently, the cost for a deviation equals 1, whereas correctly fitting events have a cost of zero assigned [CvDSW18].

While this cost function is generic, it does not use any information on data elements of events at all, an event simply is visible in a process execution log or not. This section introduces a new generic advanced cost function that aims to use all of the available information of the events in a process execution log and improve the results of conformance checking.

This approach is focusing on deviations, i.e., a missing or additional event in the process execution log to align to a possible sequence of a process model. To achieve this, acceptable values are gathered during the discovery of the process model. Acceptable values are, values for data elements that are compliant to the current process model. At the end of the section, the finding of acceptable values depending on the type of the data element is explained. When deviation is detected in an alignment, the data elements of the event after the deviation using a synchronous move are taken into consideration. If the values of the data elements result in correct values, even though an event is missing, the per-deviation cost of the preceding deviation should be reduced, since it indicates an error in logging if an event is missing in the log, or a successful repair of the process instance if an event is added to the log. Thus the following advanced cost function is defined:

Since the advanced cost function is applied on a deviation using the data elements of the succeeding event, the standard cost function is used before and the per-deviation cost of the alignment is then altered using the advanced cost function. b_m represents the base cost for a deviation, often 1. Since more than one data element can be attached to an event, the effect of specific data elements can be controlled. D contains sets of all acceptable values for each data element attached to an event. The sets of acceptable values for each data element is calculated using a process execution log as a test set or while discovering the process model from an event stream, which is explained in detail at the end of this section. The function χ is the indicator function and checks if the value for a data element, is correct by checking if the value is in a set of acceptable values D . The indicator function returns 1 if the value is present and 0 if not. The result is then multiplied by a weight, ω . This weight allows user to distribute the impact of data elements. The value of a weight is defined between $[0, 1]$ and the sum of all weights must be 1.

The base assumption here is the usage of a process execution engine, which does not allow unwanted events to be executed. If a deviation is still being detected, then there are 2 potential reasons for this deviation. Either, (a), an event is missing in the log. This could be due to an error in the log or, worse, an attack on the process execution engine, skipping events. On the other hand, if (b), an additional event is found in the process execution log, it represents a repair event or an attack as well. If the data elements of the succeeding event contain acceptable values, an error in the log or a successful repair attempt is assumed and the conformance deviation cost of the alignment reduced. If the data elements do not contain acceptable values, an attack or unsuccessful repair attempt is assumed and the cost of the alignment is not reduced. Since more than one data element can be linked to an event, each of this data elements is inspected on its own.

Implementation

Algorithm 5, shows a possible implementation of the elaborated approach. As parameters a process model, a trace and κ are needed. The parameter κ is needed to define, how many none synchronous moves in a row can be affected by the advanced cost function, since depending on the process and the process domain it may not seem feasible to reduce the per-deviation costs for lots of deviations caused by data elements of the event of the next synchronous move. At first an alignment is calculated using a standard cost function. Afterwards a loop iterates through every move of the alignment and collects κ deviation in a row. If a synchronous move is found, the advanced cost function is applied to every collected deviation and then the list is emptied, lines 5 and 5. After the loop, the cost of this alignment is calculated, through a loop collecting the cost for each move, and returned. Note that, this algorithm can be applied in an online and offline setting. For the offline setting, the alignment cost can be generated for the complete trace. In an online setting, standard online conformance checking is used [vZBH⁺17], which is calculated incrementally. Algorithm 5 is applied every time a new event is detected on the trace the event relates to.

Data Element Acquisition

To use data elements of an event in a process execution log, a distinction of the types of the data elements has to be made, since not every data element type can be used.

Input: M : Process Model with information on acceptable values for data elements
 T : A Trace, containing events of a process execution κ : maximum of consecutive deviations, ≥ 1
Result: C : Cost for the alignment of T with M .
 $alignment = \text{conformance_checking}(T)$
 $deviations = []$
for $move$ **in** $alignment$ **do**
 // iterate over every move in the alignment
 if $move.type == Deviation$ **then**
 $deviations.append(move)$
 if $len(deviations) > \kappa$ **then**
 $deviations.shift$
 end
 end
 if $move.type != Deviation$ **then**
 $advanced_cost = \text{advanced_cost_function}(move.event)$ // the event of the move
 for m **in** $deviations$ **do**
 $m.cost = advanced_cost$
 end
 $deviations = []$
 end
end
 $C = 0$
for m **in** $alignment$ **do**
 $C += m.cost$
end
return C

Algorithm 5: Finding Cost of Alignment

Numerical Data: Numerical data elements provide a numerical value which is continuous, can be compared to other numerical values and has an ordering. In the process model, for a numerical data element, an interval can be stored to check if the data element in a trace is within a certain range. This interval can be calculated for example using the interquartile range (IQR), which equals the difference between the third and first quartile. The lower bound usually then equals the first quartile minus 1.5 times the IQR, while the upper bound equals the third quartile plus 1.5 times the IQR.

In Fig. 5.6, a small example can be seen for numerical values impacting the moving costs in an alignment. The process model shows the only possible sequence is (A,B,C). Trace t shows a possible alignment, where a deviation is seen in the log, since event B is missing. For the basic per-deviation cost penalty 1 is assigned, allow one missing event and weigh each data element equally. Since 2 out of 3 data elements at event C yield acceptable values and only one deviation is used in this alignment before a synchronous move, it seems plausible that event B has taken place and a potential error has occurred while logging. Therefore the cost of this alignment drops to 0.33 instead of 1.

Categorical Data: This data element type represents data elements with a fixed number of possible values, i.e., day of the week. The XES format allows to relate a data

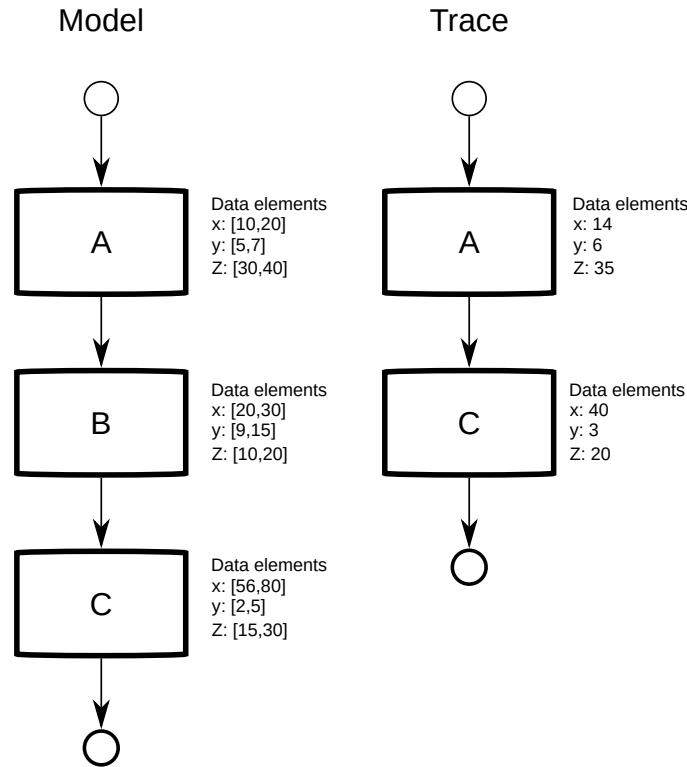


Figure 5.6: Example for Numerical Values. A range classifying acceptable values

element to a specific extension, like `lifecycle`¹, thus enabling us to distinguish between categorical data elements and arbitrary data elements. To see if a categorical data element contains a correct value, a set of possible values for this data element, is added to an event in the process model. The possible values are values that occurred as value for this data element a certain amount of times, e.g, 30%.

In Fig. 5.7, again a process model with only one possible sequence, (A,B,C), is shown, but with a different set of relevant data elements attached to event C, one of them categorical and two numerical. Trace τ features a possible alignment with a deviation for event B. Again, a basic per-deviation cost of 1 is assumed, allowing one missing event and weighing each data element equally. Both numerical data elements are not within the corresponding interval, but the categorical data element is one of the 3 possible values, therefore the alignment cost drops to 0.66 instead of 1.

Time Sequence Data: In [DRMGF14] by Dunkl et al., 3 approaches to relate a time sequence of a sensor event stream to a process model instance are introduced. The first one recording the sensor event stream separately and matching them with the process execution log afterwards. The second one, storing the time sequence in a data element and the third one splitting the sensor event stream into recurring events with a single value into the process execution log. The first approach is used here to detect the time sequence of a data element from one specific event to another.

In Fig. 5.8, the process model is enriched with the average time sequences between two events. To compare the time sequence of a process model and the time sequence of a

¹<http://www.xes-standard.org/lifecycle.xesext>

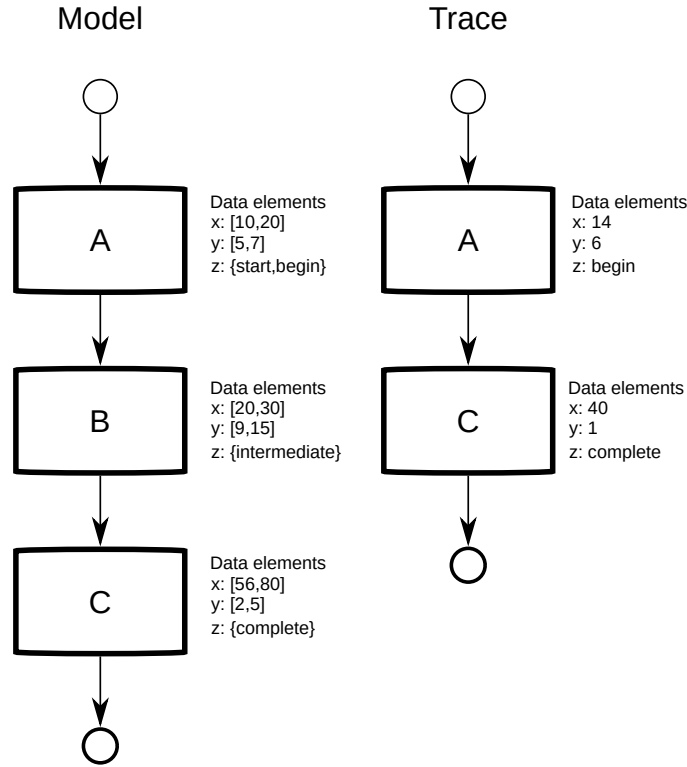


Figure 5.7: Example for Categorical Values

trace between two events, the corresponding time sequences are concatenated together, i.e., for the sequence (A,B,C), the average time sequences between A and B and, B and C are concatenated and compared to the time sequence from the trace. To compare time sequences, **Dynamic Time Warping** (DTW) is used [BC94]. DTW can cope with time sequences of different lengths and calculates an alignment between both time sequences and the cost for this alignment. If the costs are below a user specific threshold, the costs for a deviation are reduced. To find the average time sequence is calculated using **DTW Barycenter Averaging** (DBA) [PKG11]. The example in Fig. 5.8, using a basic per-deviation cost of 1, allowing one missing event and weighing each data element equally, shows that in event C, both numerical data elements are within the interval of the model and that the distance between the time sequence of data element x from the process model and the time sequence of data element x from the trace equals 1.73, which is below the threshold of 5, therefore reducing the cost of this alignment to 0.

Other data element types, e.g., arbitrary strings, are currently not supported for this method, since no relation between the data element values is known. Data element types representing a hierarchical structure, like an organizational chart, are currently being assessed and are marked for future work.

Offline and Online:

A process history allows to discover process models online, so at runtime. The information for all relevant data element types can be gathered at runtime, while time sequences from a sensor event stream can be stored online as well. Therefore the information for the advanced cost function differs, that only the currently available and processed data from

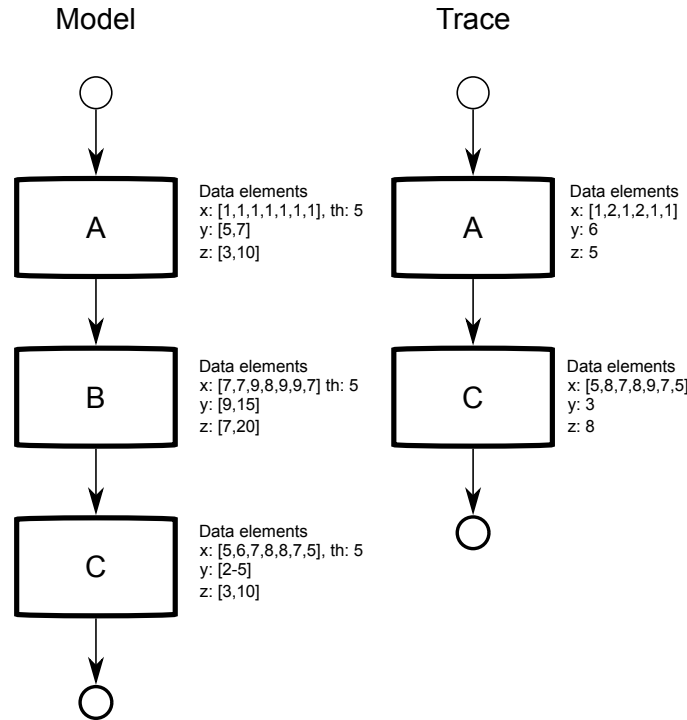


Figure 5.8: Example for Time Sequence Values. The average time sequence and the maximal allowed distance to it are stored

already executed events out of the event stream can be taken into account. The offline approach gathers the information on data element intervals, sequences and sets usually from a training set out of a process execution log without the need for a process history.

In section 5.1.2, the conformance checking method is evaluated using a real word data set from the manufacturing domain and an artificial data set to cover every possible data element type.

5.1.2 Evaluation of Extended Conformance Checking

The function and algorithm, elaborated in section 5.1.1 are prototypically implemented and evaluated based on an artificial and real-world log from the manufacturing domain. At the end of the section, the results are discussed.

The evaluation is twofold. The artificial example covers numeric and categorical data elements, while the real-world log is providing time sequence data for one data element.

Artificial Example

A test set consisting of 100 traces with data elements has been generated. The complete process model with information on the accepted values for the data elements can be seen in Fig. 5.9. For the **heart rate** and **blood pressure**, an interval is presented, while for the **status** a set of acceptable values is given.

As mentioned earlier, there are only two combinations for a trace that corresponds perfectly to the process model, i.e., either (Check General State of Health, Administer

Drugs, Check Blood Pressure, Check Heart Rate) or (Check General State of Health, Administer Drugs Check Heart Rate, Check Blood Pressure). From the data set the following acceptable values have been gathered.

- A set consisting of NOK for status in Check General State of Health.
- A set consisting of NOK for status and the values ranging from 61 to 101 for measurement in Check Heart Rate.
- A set consisting of NOK for status and the values ranging from 85.375 to 134.375 for measurement in Check Blood Pressure.
- A set consisting of NOK for status in Administer Drugs.

Figure 5.9 shows the result of Alg. 5 using the process model, κ equaling 1 and the sequences shown in the figure. Each event only contains acceptable values.

Sequence a, (Check Heart Rate, Check Blood Pressure, Check General State of Health), features a deviation, since Administer Drugs is not present in the sequence. Using the standard cost function the cost of the alignment would be 1, but since all the values are acceptable for the following event, the cost is reduced to 0 and an error in the logging is likely, since the heart rate and blood pressure show correct values.

Sequence b, (Administer Drugs, Check General State of Health), features a sequence with 2 deviations in succession. With κ set to 1, the first missing event, Administer Drugs, is not altered by the advanced cost function. This can be considered by changing the κ to 2, which reduces the cost of the alignment to 0.

Sequence c, (Administer Drugs, Check Heart Rate, Check General State of Health), features an interesting case, since again, the cost of the alignment can be reduced from 1 to 0, since all values of Administer Drugs are acceptable, but since the data element is usually constant over the first 2 events of this process, it can be argued, that this is not caused by an error in the log and the event indeed, never has been executed. This shows that the selection of relevant data elements is important and has a great impact on the results.

Real World Example

The example shown in Fig. 5.10 is from the manufacturing domain. The process model described in BPMN² is depicted without labels. The labels and decision conditions can be seen in the second column, followed by the task IDs in the third column, and the estimated times (and loop iterations) in the last column.

The whole process is executed using the cloud process execution engine (CPEE) [MRM14] on the shop floor (see Fig. 5.11) of a company specialized on prototyping and small production batches.

The process deals with the machining of a part for a gas turbine (see Fig. 5.12), and models the interplay of three machines and a stock area. MT45 is a lathe by the company EMCO with produces the parts, IRB2600 is a robot by ABB which extracts the machined parts from the lathe, moves it through a high-speed 2D optical micrometer by Keyence in order to get some production quality data. Afterwards the IRB2600 puts each part onto

²<http://www.bpmn.org/>

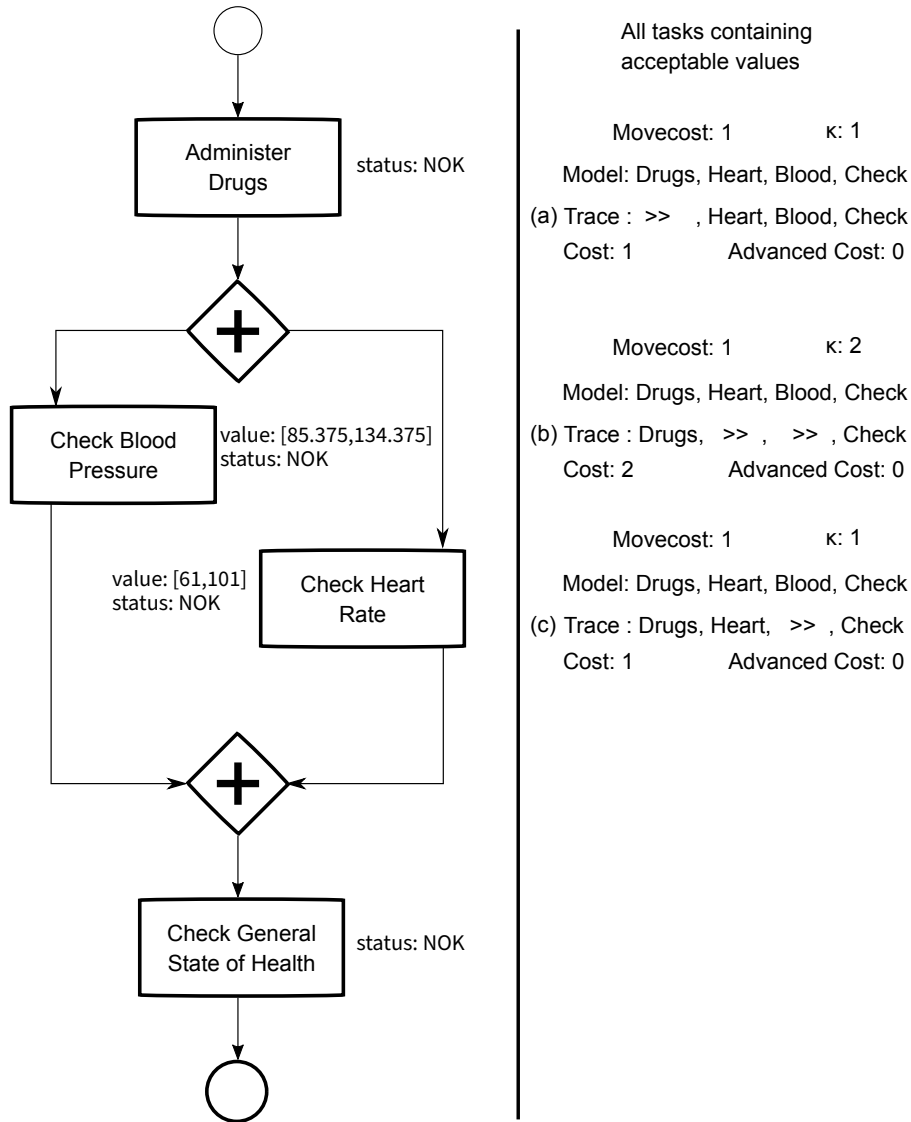


Figure 5.9: Artificial Example

an individual trays (see Fig. 5.13) in the stocking area. Each tray is labeled with a QR code, it is assumed that the trays are lined up sequentially in the stocking area, so that only the first tray has to be scanned, the QR code for each subsequent tray can then be calculated.

The process consists mostly of sub-processes which encapsulate the details of dealing with heterogeneous machine interfaces and operational safety (i.e. while the robot operates, no humans are allowed in the vicinity of the robot). The process is subject to frequent ad-hoc repair actions, as humans frequently stroll into the safety area protected by Pilz light barriers. Whenever that happens, the robot goes into an emergency stop state, that has to be acknowledged by a human. Sometimes the robot is in a position the requires manual position change by an operator in order to avoid damaging equipment, which means some steps in the process model have to be skipped or repeated.

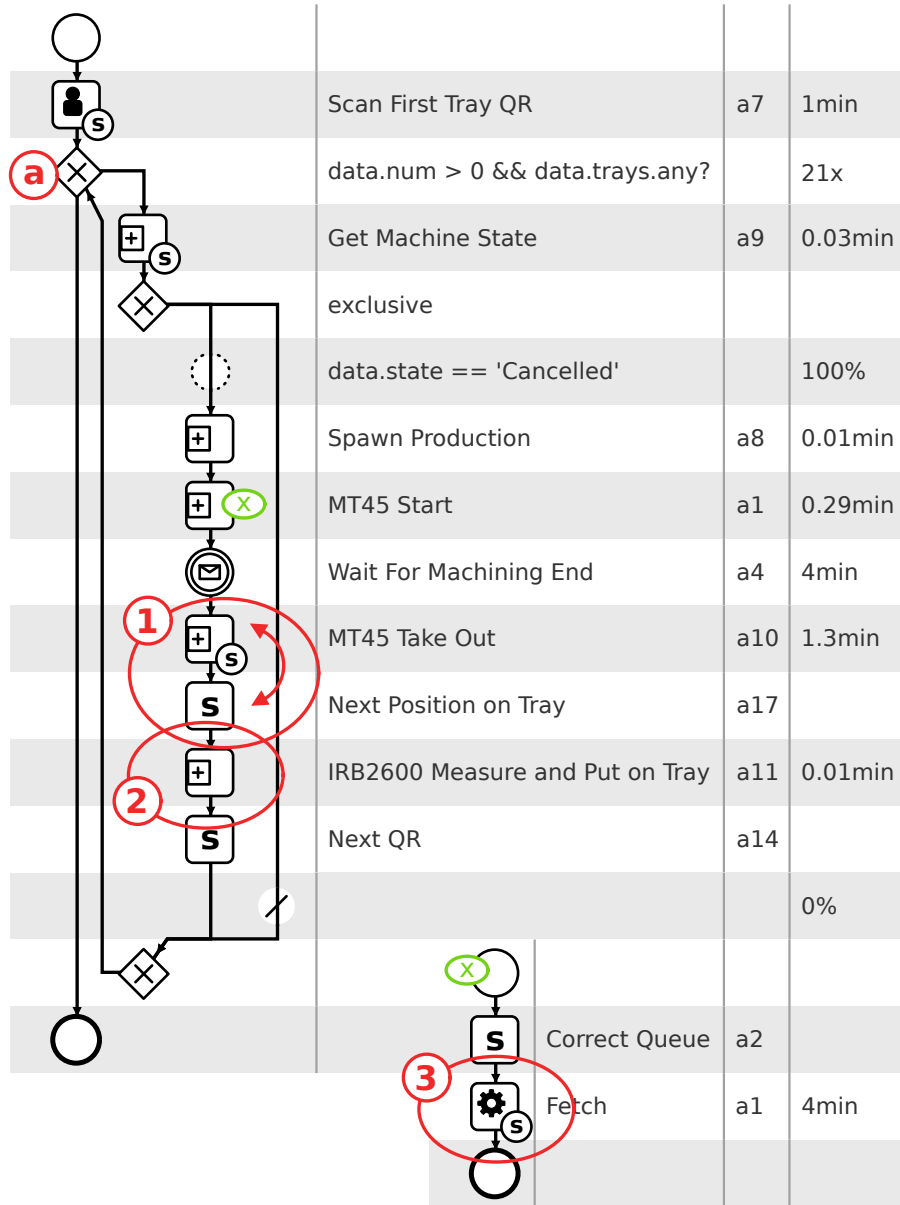


Figure 5.10: Manufacturing Example. The 'S' in the modeled tasks, shows that a script is executed by the process execution engine after a task has been completed.

The second process shown in Fig. 5.10 is part of “MT45 Start”, and deals with collecting a stream of data from 14 different sensors in the lathe while the part is produced.

When looking at the loop (a) in Fig. 5.10, out of 21 iterations, 19 had only synchronous moves. The alignments of the other 2 traces are shown in Tab. 5.1 and 5.2. κ is set to 1 and the base deviation cost is 1. The first alignment features the swap of two events, (1), while the other one the missing of an event, (2). The deviation cost for the first alignment is 2, but in GV12 MT45 Take Out and Next Position on Tray the data element `tray positions` is present. The range for all 3 values for `tray_positions` equals to $[-362.87, 437.13]$, $[-382, 21, 417.79]$, $[220, 220]$. The values of the trace are in the range of $[-112.87, -187.13]$,

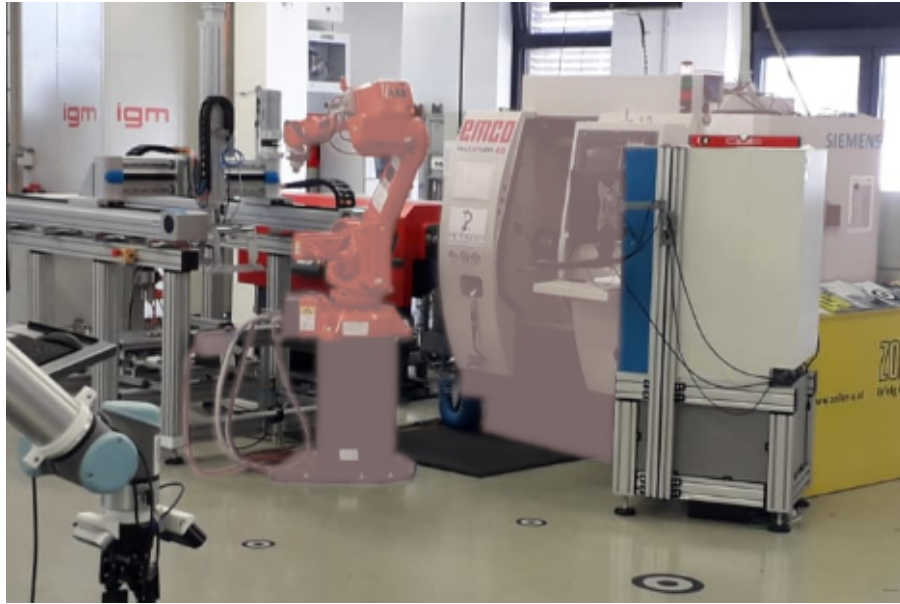


Figure 5.11: Shop Floor: Robot, Lathe, Micrometer, Stock



Figure 5.12: GV12 Part [SRM20a]



Figure 5.13: Trays in Stock Area [SRM20a]

Model	Get	Spawn	MT45 Start	Wait	Next	Take out	>>	IRB2600	QR
Trace	Get	Spawn	MT45 Start	Wait	>>	Take out	Next	IRB2600	QR

Table 5.1: Alignment of the first deviation

Model	Get	Spawn	MT45 Start	Wait	Take out	Next	IRB2600	QR
Trace	Get	Spawn	MT45 Start	Wait	Take out	Next	>>	QR

Table 5.2: Alignment of the second deviation

$[-182.21, 217.79], [220, 220]$, which are acceptable. For **GV12 IRB2600 Measure and Put on Tray**, the second deviation, features similar acceptable values, $[-212.87, 187.13]$, $[-382.21, 417.79]$, $[220.0, 220.0]$ are the acceptable values for the data element **value**. The values of the trace -112.87 , -182.21 and 220 are acceptable, hence the conformance deviation cost is reduced to 0.

For ②, only one deviation is witnessed. The acceptable values for **Next QR** are $[-362.87, 437.13]$, $[-382.21, 417.79]$ and $[220.0, 220.0]$. Since the values for the deviated trace are in between $[-112.87, 187.13], [-182.21, 217.79]$ and $[220, 220]$, the conformance deviation cost is reduced to 0.

In Fig 5.14 the average time sequence for one sensor can be seen as a red line, ③. The measurements are done in the event **Fetch**, which is done in a loop. The data set was split into a training set consisting of 10 out of the 19 traces that featured traces with a correct outcome, i.e., the part has been correctly built.

A sensor provides time sequence data, so the alignment cost can be reduced if the distance between the time sequence of the model and of the deviated trace is below a certain threshold, here 29.12. This threshold was set by a domain expert.

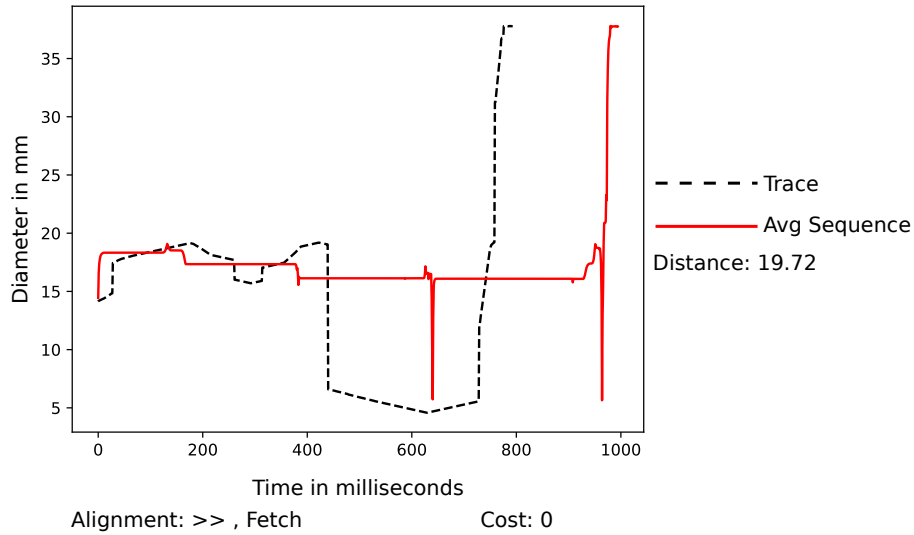


Figure 5.14: Time Sequences of Manufacturing Example. Red line is the average time sequence, dashed from one trace. The cost is reduced because the distance is below the threshold of 29.12

	Trace with Correct Parts	Trace with Faulty Part
Traces with reduced costs	50%	0%
Traces with no reduced costs	50%	100%

Table 5.3: Results Alg. 5

Because all traces out of the data set comply to the process model, the first event of the process **Correct Queue** has been randomly removed out of 20 traces.

Figure 5.14 shows the result of one trace ³ using the advanced cost function. The cost of the alignment can be reduced to 0, because the distance between both time sequences is 19.72 below the threshold of 29.12. This threshold is set by a domain expert.

The 17 traces without missing events, have been aligned perfectly with an alignment cost of 0. Out of the 20 traces, 8 traces resulted in an accepted outcome of the process, while the other 12 traces yielded faulty parts. Out of the 8 traces, the alignment cost of 4 traces has been correctly reduced to 0, since even though the **Fetch** event is missing, the time sequence is below the threshold and the part is correct.

Out of the 12 traces with faulty parts, no alignment costs have been reduced at all, since the time sequence difference was too big. This leads to believe that the error is in the execution and not in the logging, therefore the alignment cost using the standard function is correct.

Discussion

The results of the artificial and the real world example showed interesting results. The best results are achieved using data elements that always yield the same values in every process instance. For example, in the manufacturing domain, the dimensions of the produced part are always the same, thus the interval for this data elements can be calculated easily. The same can be observed while using time sequence data elements. Since the time sequences are fairly similar of all process instances. On the other hand, i.e., if the values for the data elements vary and do not follow a pattern, the advanced cost function cannot be applied. For example in the finance domain in a loan approval process, the amount of the loan can vary greatly and the different instances are not related.

This leads to an important aspect of the advanced cost function, the selection of relevant data elements of the events. This can be seen in the artificial example.

As mentioned before, data element that vary greatly are problematic, since acceptable values would contain a great range. This would result in a reduced alignment cost, since almost all values are acceptable for such a data element even though it is faulty. Therefore experts have to select viable data elements to apply the advanced cost function onto.

The value of a data element at a certain amount of time, always leaves some room for interpretation. Time sequence on the other hand shows the full history of a data element. For example, the blood pressure of a patient should be elevated for a certain time span instead of one measurement. It can be seen that the advanced cost function on the real world data set assigns lower alignment cost values for good parts and does not reduce the

³79917b2a-5bac-4074-8690-4e4e5cdf0b8f

alignment cost for faulty parts for 50% of the traces. This leads to the assumption, that the costs be calculated best using time sequence data.

The algorithms in this section presented a novel approach to adjust the costs of mismatches in an alignment based on an advanced cost function. The next section focuses on the next area of Figure 5.2, semantic deviations.

5.2 Temporal Perspective

while the previous section presented new approaches to adjust the cost of structural deviations detected via conformance checking methods, this section focuses on the semantic deviations [MdLRvdA16a] (cf. Figure 5.15), mainly *temporal deviations*.

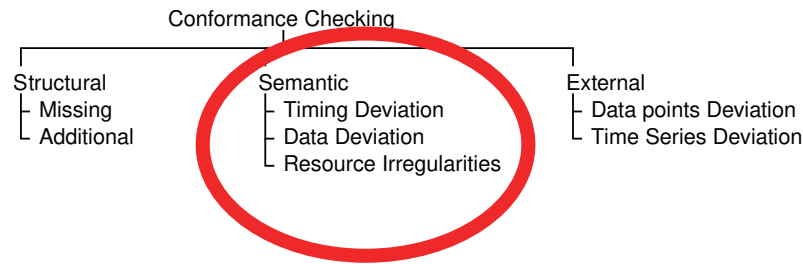


Figure 5.15: Conformance Checking Types and Affected Artifacts. This section focuses on the circled area.

Temporal information in a process can be defined in two different ways: (a) the time between start events of two subsequent tasks (*temporal distance*) and (b) the time between start and end events of a single task (*task duration*) [PLCR19]. In general, logs for mining purposes are either generated by extracting data from a specialized information system, or alternatively generated by a process execution engine. Most available data sets only contain end events. Thus temporal distance is the prevalent definition, while some process managements systems produce logs with start and end events and thus allow for the more concise task duration point of view.

Temporal deviations might occur for the following reasons:

- Reduced duration: A task was not executed properly because, e.g., a machine failed or information was missing.
- Increased duration: Preconditions for the tasks have not been met, or hidden dependencies between tasks exist.

This section aims at finding and quantifying such temporal deviations. By contrast to existing work [MdLRvdA16a], the temporal information is not available in the form of temporal constraints, but is determined based on an input process execution log. The result is a *temporal profile*. The input process model is infused with the temporal profile. The goal of this work is to determine temporal deviations of an (ongoing) process event stream with the temporal profile during runtime (online). A *process event stream* contains events reflecting the execution of process instances. Instead of a protocol of finished process instances, the events of all process instances currently being executed are being put into a stream. An event stream contrary to a process execution log is infinite.

Another important feature of this approach is to propose a user-adjustable semantic quantification (in the form of an annotation to the process model) in order to quantify the significance of temporal deviations for a certain task. This way, it can be expressed that exceeding the limit of a task duration can be more severe for some events (or tasks) and even affect succeeding events.

Two real-world data sets are used for the evaluation of the approach. The first one is the BPI Challenge 2012, since it meets the criteria of providing the start and end time of at least some events of the process execution log [Van12]. The other data set is from the manufacturing domain and features the production of industry parts where the whole production process is managed and enacted by a process execution engine.

Section 5.2.1 provides the algorithm for infusing process model with a temporal profile. Moreover, Sec. 5.2.1 introduces the cost function for temporal deviations and the algorithm incorporating it. The contribution is then evaluated and discussed in Section 5.2.2.

5.2.1 Temporal Conformance Checking

This section infuses process models with temporal profiles as input for temporal conformance checking using a cost function to quantify temporal deviations.

Temporal Profile Generation

The basic idea of temporal conformance checking is to infuse a process model with a temporal profile and then to conduct conformance checking using a cost function that quantifies temporal deviations between the temporal profile and an event stream of interest. The input for temporal conformance checking hence comprises a process model and (i) a process log in an offline setting or (b) an event stream in a runtime (online) setting. For (i) the process log can be split into a training set for calculating the temporal profile and a test set for temporal conformance checking. The evaluation will show both, (i) offline and (ii) online settings

The temporal profile captures task duration and temporal distance between events/tasks. Both are explained in the following and sketch some scenarios. Algorithm 6 calculates a temporal profile from a process model and a process execution log. An example for a process model infused with a temporal profile is provided in Sec. 5.2.1.

Task Duration: To compare the attached event data between process execution logs in the XES format, standard extensions are available, for example, the name, the timestamp, and the lifecycle of an event. The lifecycle data element of an event reflects its current status. For temporal conformance checking, start and end lifecycle event are needed to calculate the complete task duration by calculating the difference between these two timestamps (cf. Fig. 5.16 (b)).

Temporal Distance: The temporal distance determines the time after an event is completed and before a new event starts. For this, the events have to contain again a data element describing its lifecycle, i.e., start and complete. The notation $|AB|$ is used for describing the temporal distance between event A and B. While in a strict sequence of events, like events A and B in Fig. 5.16 (a), the computation of the temporal distance is trivial, some interesting occurrences can be witnessed if events are executed in parallel, i.e., event C and D in Fig. 5.17.

Event B has to be completed before either event C or D can be started. This leads to 4 possible temporal distances, namely $|BC|$, $|BD|$, $|CD|$, and, $|DC|$. Both, C and D have to be finished before event E can be performed, thus the distances $|DE|$ and $|CE|$ emerge as well.

Algorithm 6 calculates the mean and standard deviation for all observed temporal distances and stores them. Infrequent distances can be filtered out using a threshold value.

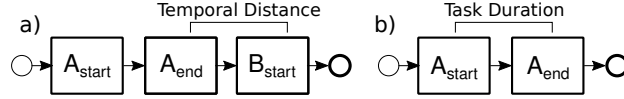


Figure 5.16: Temporal Profile – Example.

For this, Alg. 6 starts with creating two hash tables [CLRS09] to save the time distances, one for task duration and one for temporal distances. Afterwards, each event of every process instance is parsed. If a start event is detected, the temporal distance is calculated to the last detected end event and the timestamp is saved for the task duration. On the other hand, if an end event is detected, the task duration is calculated and the event is saved as the currently last finished event for the temporal distance. Lines 6 and 6 determine the mean and standard deviation and filter out infrequent temporal distances.

Temporal Conformance Checking

After temporal profile is calculated, temporal conformance checking can be performed, either online at runtime on an event stream or offline on a process execution log.

The **z-score** [CA86] is used to determine the distance of new observations to the gathered data sets from the preparation phase. Since the data for the data sets has been collected in a complete test set, it can therefore be argued that the data set is complete, which enables the use of the z-score. The z-score is defined as follows:

$$z = \left| \frac{x - \mu}{\sigma} \right|$$

Thus, the deviation is calculated by subtracting the mean of all time distances for an event from the new observation and divides it by standard deviation of all time distances for one event. If the z-score exceeds a specified threshold, a deviation is detected. Certain tasks allow a greater deviation while other tasks demand to be very precise. The z-score determines how many standard deviations the observation is distant to the mean. Thus a list of thresholds is used, containing a threshold for every task in the process model to reflect the needs for adjustable outlier detection.

To use this z-score for quantifying the cost of a temporal deviation, a temporal deviation cost is introduced as follows:

The end cost for an alignment calculated by temporal deviation conformance checking is the sum of all costs of structural movements using standard conformance checking, i.e., log and model moves, and of all costs of temporal deviations using the temporal deviation cost function.

Algorithm 7 shows a prototypical implementation for temporal conformance checking at runtime. Line 7 starts a hash table for all currently saved process instances. The parameter *T SIZE* defines how many process instances can be stored at the same time. Since the

Input: M : A process model,
 L : A process execution log, containing traces of process instances, κ : threshold for filtering infrequent time distances
Result: M : A process model containing information of time distances

```

 $td = \text{dict}()$  // Hash Map of Execution time duration
 $inter\_td = \text{dict}()$  // Hash Map of Inter Time distances
for  $trace$  in  $L$  do
    // iterate over every process instance in the log
     $current\_starts = \text{dict}()$  // Hash Map for execution time duration
     $last\_end = \text{None}$ 
    for  $event$  in  $trace$  do
        // iterate over every event of trace
        //  $lc() == \text{lifecycle of event}$ ,  $ts() == \text{timestamp of event}$ 
        if  $event.lc() == 'start'$  then
             $current\_starts[event.name] = event.ts()$ 
            if  $last\_end \neq \text{None}$  then
                if  $inter\_td[last\_end.name+event.name] == \text{None}$  then
                     $inter\_td[last\_end.name+event.name] = \text{list}()$ 
                end
                 $inter\_td[last\_end.name+event.name].append(last\_end.ts()-event.ts())$ 
            end
        end
        if  $event.lc() == 'complete'$  then
            if  $td[event.name] == \text{None}$  then
                 $td[event.name] = \text{list}()$ 
            end
             $td[event.name].append(event.ts()-current\_starts[event.name].ts())$ 
             $current\_starts.remove(event.name)$ 
             $last\_end = event$ 
        end
    end
end
 $stats = \text{dict}()$  // Hash Map of all means and standard deviations of time distances
for  $key$  in  $td$  do
     $stats[key] = (\text{mean}(td[key]), \text{stddev}(td[key]))$ 
end
for  $key$  in  $inter\_td$  do
    if  $\text{len}(inter\_td[key]) \geq \kappa$  then
         $stats[key] = (\text{mean}(inter\_td[key]), \text{stddev}(inter\_td[key]))$ 
    end
end
 $M.add(stats)$  // Infusing Process Model with Hash Map of time distances
return  $M$ 

```

Alg. 6. Temporal Profile Generation

algorithms are dealing with possibly infinite process instances, only a fixed number can be stored. If the hash table is full, an older process instance is removed from the table. There are many classifiers possible for determining the to be removed process instance. This approach opted for the oldest process instance to be removed, line 7. Lines 7 to 7 create the necessary variables for each instance.

In line 7 the alignment cost for standard conformance checking is calculated [vZBH⁺17]. The hash map *unfinished_events* saves all timestamps of starting events. Every time a starting event is detected, the timestamp is added. If the related end event is detected, line 7, the timestamp is removed from the hash map. For each end event, the temporal deviation cost is calculated based on the duration of the event, line 7. For every detected

Definition 7 (Cost Function for Temporal Deviations).

$$\text{temporal_deviation_cost_function}(x, M) = \begin{cases} 0, & \text{if no mean}(M_x) \text{ available || } z\text{-score}(x) \leq \kappa_{M_x} \\ \omega_{M_x} * \phi * z\text{-score}(x), & \text{otherwise} \end{cases}$$

Let x be a time distance, M a process model containing information of time distances, κ a threshold defined between 0 and ∞ and M_x the data set of related time distances, containing the mean and standard deviation and a related κ -threshold. If a mean is available and the z-score is smaller or equals κ , this function yields 0. Otherwise let ϕ any number between 0 and inf to adjust the impact of a temporal deviation in general and let ω_{M_x} be a weight between 0 and ∞ for the related event to adjust the impact of specific events. The function yields the temporal deviation cost for observations with a z-score greater than the threshold κ .

start event, the temporal deviation cost for inter event time distance is calculated, line 7.

If an event is still being executed and its task duration is already greater than the average time for this task, the temporal deviation cost is calculated using the time duration between the moment the last event is detected in the event stream of this process instance and the starting event, line 7.

Illustrating Example

As can be seen in Fig. 5.17, the process model has been infused with a temporal profile capturing temporal distances and task durations on average and with the standard deviation.

Assume the following traces t_1 and t_2 to appear in an event stream of interest:

$$t_1 = ((A_{start}, 0), (A_{end}, 19), (B_{start}, 29))$$

$$t_2 = ((A_{start}, 0), (A_{end}, 20), (B_{start}, 23), (C_{start}, 24), (C_{end}, 28), (B_{end}, 29)).$$

Note that in order to illustrate the run through of Alg. 6 and Alg. 7, the events are listed with their time spent related to the start of the process instance in seconds.

The ω_{event} is set for A and B to 1 and for $|AB|$ to 2 to represent a more severe deviation, ϕ to 1 for both instances and κ_{event} to 2 for the task duration of B and to 3 for all other distances. The events in t_1 are in the correct order, so there are no costs for aligning this trace to the process model. The z-score for event A is calculated when A_{end} is detected, which yields 0.25. Since 0.25 is smaller than 3 (κ), no temporal deviation cost is added. When B_{start} is detected, the temporal deviation cost is calculated based on the temporal distance of 10. This yields a z-score of 14, which is greater than κ , increasing the cost for this trace to 28, since ω is set to 2 for this distance. Assume that 7 seconds passed after B_{start} has been performed and the algorithm has been executed. Since that is greater than the average execution time of 6 seconds of event B , the temporal deviation cost is calculated, which yields a z-score of 2. Because κ is set to 2 for the task duration of B , the cost value is increased by 6. The current cost of this trace is therefore 34.

t_2 shows an deviating structure, since the execution of C has been executed before event B finished, which results in move in the alignment and by using the default conformance checking cost function yields costs of 1. The execution time duration of all events is fitting the process model, as well as the temporal distances. There are no recordings for the

Input: M : Process Model with information on time distances
 ES : An event stream, sending events related to model M .
 $TSIZE$: Maximum Number of available process instances that can be stored.
 κ_{event} a list of thresholds the maximum allowed z-score per event
 ω_{event} a list of weights for the results of temporal deviation cost function per event
 ϕ : general cost modifier for temporal deviations
Result: C : Cost for last alignment of ES .

```

// temporal_deviation_cost_function(x) == tc(x)
traces = dict()
for e in ES do
  if e.trace not in traces then
    if len(traces) ≥ TSIZE then
      traces.remove_oldest()
    end
    traces[e.trace] = dict()
    traces[e.trace]['cost_time'] = 0
    traces[e.trace]['preceding_event'] = None
    traces[e.trace]['unfinished_events'] = dict()
    traces[e.trace]['trace'] = list()
  end
  t = traces[e.trace]['trace']
  t.append(e)
  traces[e.trace]['cost_structural'] = online_conformance_checking(t,e)
  if e.lc() == 'complete' then
    if tc(e,M) >  $\kappa$  then
      traces[e.trace]['cost_time'] += tc(e,  $\omega_e$ ,  $\phi$ )
    end
    traces[e.trace]['preceding_event'] = e
    traces[e.trace]['unfinished_event'].remove(e.name)
  end
  if e.lc() == 'start' then
    if traces[e.trace]['preceding_event'] != None and
       tc(|traces[e.trace]['preceding_event']|e|) >  $\kappa_e$  then
      traces[e.trace]['cost_time'] += tc(|traces[e.trace]['preceding_event']|e|,  $\omega_e$ ,  $\phi$ )
    end
    traces[e.trace]['unfinished_events'][e.name] = e.ts()
  end
end
end
C = traces[e.trace]['cost_structural'] + traces[e.trace]['cost_time']
return C

```

Alg. 7. Finding Cost of Alignment

temporal distances $|AC|$ and $|CB|$, thus the temporal deviation costs cannot be calculated for these distances.

5.2.2 Evaluation of Temporal Conformance Checking

Algorithms 6 and 7, together with the cost function for temporal deviations (cf. Def. 7) are evaluated based on two real-world data sets. The one is the BPI Challenge 2012

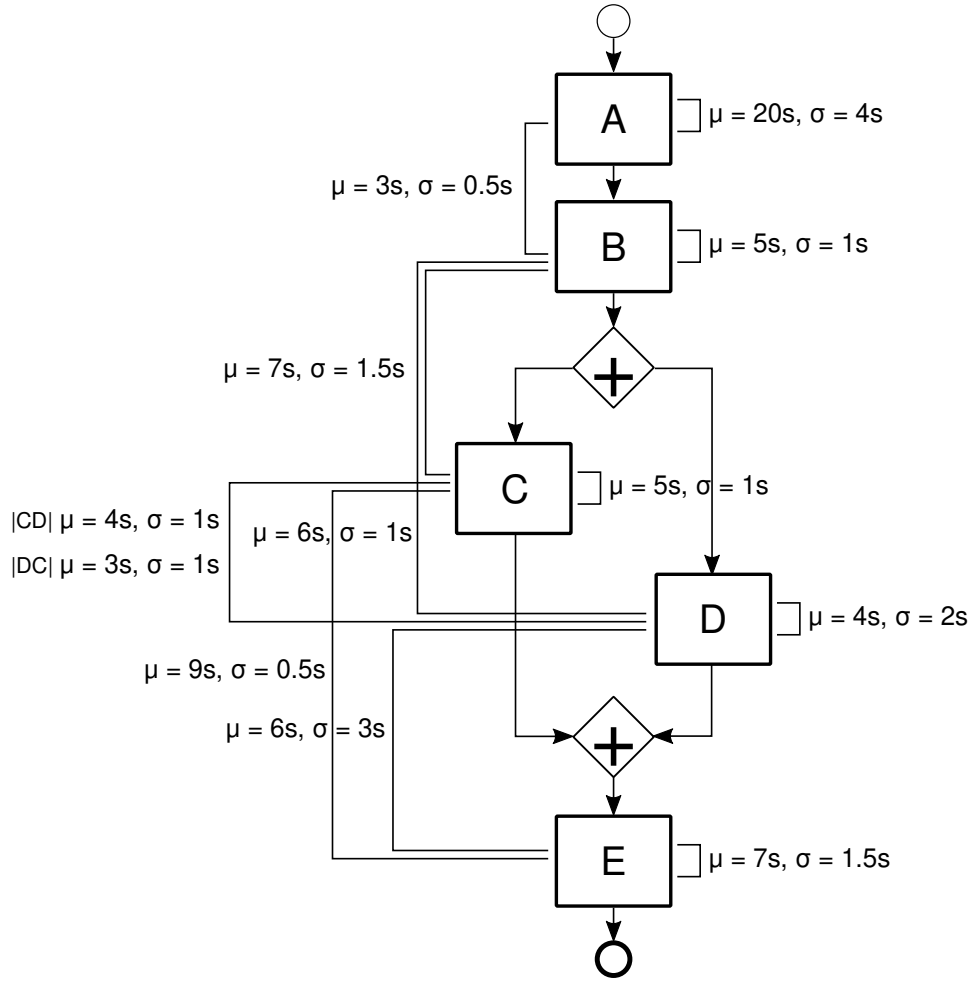


Figure 5.17: Process Model Infused with Temporal Profile – Example

(BPIC2012 for short) [Van12], which features a process from the financial domain. This log has been chosen, since it provides lifecycle attributes for at least a few number of events to calculate the temporal distances. The feasibility of the approach is evaluated using this log.

Since no additional knowledge of the log is present to evaluate the results of temporal deviation conformance checking, a second real-world is presented in the evaluation. The second log [SRMM] features a business process in the manufacturing domain⁴ and experts are evaluating the results afterwards to assess the applicability of the approach. A prototypical implementation of the temporal profile can also be found in the `pm4py` framework [BvZvdA19].

5.2.3 Financial Example

The data set from the BPI Challenge 2012 contains 262200 events from 13087 process instances. Since only the process execution log is given by this data set, the evaluation

⁴http://gruppe.wst.univie.ac.at/data/manufacturing_log.zip

Name	Profile Size	μ	σ	Min	Max
W_Completeren aanvraag	18562	640.03	5883.48	0.77	244731.43
W_Nabellen offertes	18975	560.58	7302.64	0.95	243191.22
W_Valideren aanvraag	6493	1268.71	6098.85	1.1	238256.25
W_Afhandelen leads	4864	1012.62	9905.82	0.67	243739.82
W_Nabellen incomplete dossiers	9574	771.07	8052.62	1.03	239878.67
W_Beoordelen fraude	211	73.77	640.81	0.77	9240.84

Table 5.4: BPIC 2012: Task Duration in seconds for all 6 events of 3271 process instances

mainly focuses on calculating the additional temporal deviation costs for an alignment without the costs of structural conformance checking and assign ω_x to 1 for every event. The events contain data elements describing the values for a financial transaction as well as timestamps. A mandatory requirement for the elaborated approach of this section, is the availability of lifecycle attributes of an event.

Out of the 23 different tasks in the business process, only 6 tasks have a lifecycle logged with a start and an end event. This can be explained, because the process consists of 3 inter twined sub-processes, but only sub-process W contains the start and end event of activities, while the other two A and O do not.

The data set is analyzed as a whole instead of breaking it up into 3 different processes, because these sub-processes are inter-twined, therefore do some temporal distances appear from sub-processes without start events. The task duration does not change if other events happen during the execution. Since no event stream of the data set is available, the data set is split into a process execution log consisting of the first 80% of process instances and an event stream constructed using the other 20% of the process instances, a similar approach to machine learning algorithms [Alp20].

For Algorithm 6, the first 10469 process instances are used as a set to gather the temporal profile.

As can be seen in Tab. 5.4, the task duration varies to a great extent. Often the duration of an event takes seconds, other times the duration spans several days. This can be attributed to the fact, that these events are inter-twined with events from the other sub-processes. Hence the task duration time of these events is depending on the time of the other events. The duration cannot be calculated for the other sub processes, since only end events are found in the process execution log.

As sub-process W contain start events, temporal distances can be determined. κ is set to 20 to filter out distances, that only have been detected like 2% of the time.

The remaining temporal distances can be seen in Tab. 5.5. Again, a wide range of time distances between the events is detected. This is reasonable, because the task duration varies to a great extent as well.

Algorithm 7 is applied to the test set consisting of the remaining traces in the log. Even though the algorithm has been designed for online execution, it is still possible to calculate the temporal deviations in an offline setting. To achieve this all events have been collected and have been sorted according to their timestamp. Each of these events has then been injected into an event stream with random intervals, but still using their original timestamp for the calculation. κ_{event} has been set to 3 for all events, ϕ and all ω_x to 1.

Out of 12650 task duration, 12607 events yielded a z-score below κ , but 43 yield a higher score. The maximum z-score of 3620.9 has been found in process instance 207263. As can

Name	Profile Size	μ	σ	Min	Max
A_PREACCEPTEDW_Completeren aanvraag	3819	22875.3	33214.99	3.55	156982.87
W_Completeren aanvraagW_Nabellen offertes	4019	270672.33	308357.9	6.61	1206235.29
W_Nabellen offertesW_Nabellen offertes	14955	258339.44	279374.88	1.4	2572740.11
W_Nabellen offertesW_Valideren aanvraag	2640	233294.08	159396.81	7.2	617152.84
W_Completeren aanvraagW_Completeren aanvraag	12686	79836.59	228186.05	1.47	2583000.9
W_Valideren aanvraagW_Valideren aanvraag	2402	39583.82	112968.6	1.95	1289690.02
A_PARTLYSUBMITTEDW_Afhandelen leads	3905	18670.85	30043.74	11.57	151792.25
W_Afhandelen leadsW_Completeren aanvraag	2064	12132.28	20071.87	6.76	159723.98
W_Valideren aanvraagW_Nabellen incomplete dossiers	1764	5108.96	8359.51	4.62	147659.13
W_Nabellen incomplete dossiersW_Nabellen incomplete dossiers	7809	61731.99	112940.21	1.54	1117225.91
W_Nabellen incomplete dossiersW_Valideren aanvraag	1421	34776.54	75051.87	9.97	614586.68
W_Afhandelen leadsW_Afhandelen leads	940	3669.69	12485.41	1.46	160780.79


Table 5.5: BPIC 2012: Temporal Distance in Seconds of the first 10469 process instances with κ set to 200


be seen in Tab. 5.4, **W_Beoordelen fraude** is the event with a small standard deviation. In this process instance the task is started at 09:00 on a Thursday and finished the next day at 06:15. Thus a holiday does not seem plausible and it is likely that something happened during this task. The advantage of using this algorithm online would be the possibility to examine the process instance immediately when enough time has past after the starting event of this task. No process instance showed more than 2 deviations.

Out of 12654 possible temporal distances, 12395 do not deviate and 259 do. The maximum deviations of one process instance is 5, detected in 2 process instances. Without domain experts, it is difficult to classify the severeness of these deviations.

5.2.4 Manufacturing Example

The first example showed that the algorithm is returning good results with no knowledge of the underlying process. For the manufacturing data set⁵, a process model with different ω_x and κ_x is provided. An expert has been assessing the results.

Figure 5.18 shows the manufacturing of a part⁶. The main process (tasks a1..15) shows the interaction between different machines (EMCO MT45 Turning Machine, ABB IRB2600 Robot, Keyence Optical Measurement Machine) during production, while the sub-process  is spawned for every single produced part, and tracks its full production lifecycle. Every task has an ω assigned, values of 1 signify normal (high) importance. Values of 0 signify that deviations can be ignored. In addition to that, a κ is assigned to each task, representing how many standard deviations an observation can be distant to the mean of a distance. Often 3 is used as a default value to detect outliers [CA86]. A higher value allows for greater distant distances, while a smaller value implies, that the distance has to be closer to the mean.

The  sub-process is forked, i.e., the main process does not wait for it to finish, but executes in parallel. The duration between the b1 start event and the b1 end event should always be identical to the duration between the a8 start event and the a4 end event. For $a1 : \omega = 0$, as $a1$ forks a sub-process (without waiting for it to end). Hence its duration is negligible.

⁵http://gruppe.wst.univie.ac.at/data/manufacturing_log.zip

⁶Note that the tasks IDs a1 to a15, and b1 to b4, are neither continuous nor in sequence as this version of the process is the result of several redesigns

Every occurrence of $\omega = 1$ is attributed to actual machining of the part, which is expected to show small deviations. When the MT45 turning machine closes its door and starts machining, only two cases can lead to a deviation: A power failure or if raw material or the machining tool breaks. Both of these cases are unlikely. Especially the latter is important. If a tool or the raw material breaks, two things can happen: (1) the machine triggers an emergency stop if any damage due to flying metal parts is detected, or (2) the machining time is reduced, as there is no longer contact between machining tool and raw material. Thus no friction occurs⁷.

Thus, a first perceived application of temporal conformance checking is to trigger an emergency stop of the machines.

All tasks with the prefix IRB2600 are robot tasks (a10, a11, a12, a13, a14, a15). The robot takes the finished parts out of the MT45 Turning Machine. This can partly happen in parallel to the machining. As soon as the robot leaves the confines of the MT45, the machine can start producing the next part. While the MT45 is producing the next part, the robot (a12, b2) scans the part with the help of a Optical Measurement Machine, and (a15) puts it on a tray which, in turn, is placed on pallet with 50 other trays. All of the IRB2600 tasks are expected to show small deviations, except for a15. As all the trays are on a different position on the pallet, each operation should have a slightly different duration, thus yielding the highest standard deviation of all tasks. All robot tasks are also prone to extreme outliers, as the robot is subject to an industrial safety mechanisms: whenever someone accidentally walks close to the robot, a full emergency stop is triggered to avoid injuries. This can indeed be observed in the data multiple times. Afterwards it is not always trivial to restart the robot as sometimes it has to manually be moved into a defined state, before the process can be continued.

As a second application, temporal conformance checking can be used to automatically determine the significance of cases, in order to notify personnel for solving the situation.

Table 5.6 shows selected durations of different tasks. Temporal conformance checking proves efficient in pointing out small deviations in the manufacturing process, instantaneously at runtime. For the task duration, 27 deviations of 1373 distances and for the temporal distance, 47 deviations of 1334 distances have been detected. At one point the raw material was running out. Thus the resulting part was missing some of its mass. Due to the reduced weight, slightly different timings can be accurately pointed out in the log. While this is probably also possible through other means (e.g., analyzing the deviations in data from the measurement), the runtime temporal deviations provide much faster and more universally applicable feedback.

Threats to validity: Even though real world data sets are used in this evaluation there are still potential problems. One concern is the volume and velocity of data in the event stream. While both of these data sets are rather small and can easily fit into the memory of a modern computer, there could be performance issues when dealing with a real infinite event stream. Another important aspect is the need for an expert to receive satisfying results. Even though deviations have been detected in the finance example, it is hard to argue automatically if these deviations are a real concern for the process instance or are negligible.

This section presented algorithms to identify temporal deviations and perform conformance checking on a semantic perspective. The next section focuses on the last area of

⁷This can also be confirmed through lower power usage.

Name	Profile Size	μ	σ	Min	Max
IRB2600 Grip	30	40.66	6.3	37.32	74.03
IRB2600 Extract	30	23.17	1.43	21.21	28.85
IRB2600 Portal to Door	30	12.22	6.28	9.48	45.91
IRB2600 Door to GS	30	12.47	0.56	11.48	13.85
IRB2600 GS to Take	30	14.78	0.47	13.96	15.81
IRB2600 Take to GS	30	10.5	1.1	9.69	15.73
IRB2600 GS to Door	30	11.49	0.7	10.1	13.12
IRB2600 Door to Scanner	30	13.75	1.62	11.19	18.2
IRB2600 Scan	30	29.59	1.82	26.08	32.59
IRB2600 Scanner to Door	30	12.89	0.98	11.19	14.84
IRB2600 Door to Portal	30	10.96	1.47	9.56	18.03
IRB2600 Unload to Tray	30	20.86	9.6	17.4	69.16

Table 5.6: Manufacturing: Task Duration in Seconds of first 30 process instances in the data set.

conformance checking, external data analysis and presents an approach to explain the root cause of a concept drift using external sensor data streams.

5.3 Discover and Explain Concept Drifts based on external data sources

This section focuses on external data used in conformance checking as can be seen in Figure 5.19.

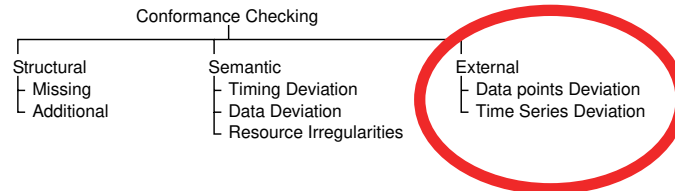


Figure 5.19: Conformance Checking Types and Affected Artifacts. This section focuses on the circled area.

“World-class organizations leverage business process change as a means to improve performance, reduce costs, and increase profitability” [The19]. Companies can react by adapting their business process to the changing requirements at a large scale, e.g., new regulations, and at a smaller scale, e.g., deviations in sensor streams in manufacturing or medicine. In any case, adaptations of the process logic result in a so called *concept drift* [WK96].

When adapting business processes, a concept drift might be known in case of explicitly defined and applied process changes, but also unknown and “only” recorded in so called *process event logs* that store information on business process execution in an event-based manner. If the process execution events are continuously collected during runtime, a *process event stream*. Existing techniques detect concept drifts from process event logs in

an offline manner (ex post) based on process execution logs [BVDAZP14] or in an online way based on *process event streams* [vZvDvdA18, MBCS13, SRM18], i.e., during runtime as the processes are executed. Online concept drift detection can be crucial to react on process changes in time. However, approaches to analyze and identify the reason why a concept drift happened, i.e., its *root cause*, are missing although knowing the root cause contributes to, e.g., optimizing future occurrences of similar concept drifts.

Hence the basic question is how to identify and analyze the root cause for concept drifts at runtime. Several examples suggest that data from IoT devices, i.e., external sources such as sensors can influence the execution behavior of a process. Temperature, for example, might cause exceptions in logistics processes [BFN⁺19]. Variations in parameters might indicate the quality of products in manufacturing [KHP⁺19, EFMR19]. The data emitted by sensors is called *sensor event streams* and is captured externally, i.e., outside the process execution [MPO⁺19]. Sensor event streams constitute *time sequence data* [KHP⁺19]. Informally, a time sequence holds quantitative, time-stamped data.

In order to facilitate root cause analysis for concept drifts, this chapter focuses on aligning drifts in sensor event streams to associated process instances and helping domain experts to assess root causes and thus propose concept drifts.

The approach takes the methods from the previous chapters as input. The process history holds an ordered sequence of process models that have been mined online and are connected to per-instance sensor event streams. These sensor event streams are time sequences, and the deviations between the streams of different instances of each model are determined using *dynamic time warping (DTW)*. DTW calculates the distance between two time sequences. The challenge is to interpret the drifts in the sensor event streams to identify future concept drifts in the process model (in contrast to [SRM18], which deals with the identification of concept drifts ex-post). The approach was implemented for a real-world IoT application from the manufacturing domain and a data set was gathered that is used to evaluate the approach. Specifically, it is shown how the results of the analysis support domain experts in understanding why a drift happened (root cause) and to learn what can be done to efficiently deal with it. The objective is therefore to evaluate the applicability of this approach in finding the cause for a concept drift, to evaluate the performance of this approach, i.e., how many reasons can be correctly detected in which time and to give information on how to adapt the current process model to the current situation.

This section is outlined as follows: In Section 5.3.1, a running example as well as preliminaries are introduced. Section 5.3.2 features two algorithms to determine drifts in event streams from external sources. Section 5.3.3 evaluates these algorithms based on a real world IoT application. The results and impressions are discussed in section 5.3.4.

5.3.1 Fundamentals of Dynamic Time Warping

Figure 5.20(a) shows the process model of a medical round for a patient of a health care facility. This model represents the current care plan for one specific patient. The general health status of a patient is checked, the blood pressure is measured, and drugs are administered. During runtime process instances are created and executed based on the process model. The execution information is stored in a process event log. Assume that a concept drift occurs, which results in the process model depicted in Fig. 5.20(b), i.e., an additional hydration check is added in parallel to checking the blood pressure. Another

drift could be detected in the data elements of a process instance, for example, the task “Blood Pressure” is in (b) done by nurses, while it has been done in (a) by medical doctors. Existing approaches [MBCS13, SRM18, vZvDvdA18, MDRO17] enable drift detection, but do not explain why the drift happened in the first place.

Unlike process data such as resource or patient age, typically, the temperature and humidity of the patient’s room are constantly monitored by external sensors. The sensors produce event streams which consist of data points representing a single measurement. These measurements are typically not stored in a process execution log, but in a different database, since the tasks are not directly linked to any process data. It is investigated whether and how such sensor event streams can be exploited in order to analyze and explain why a concept drift happened.

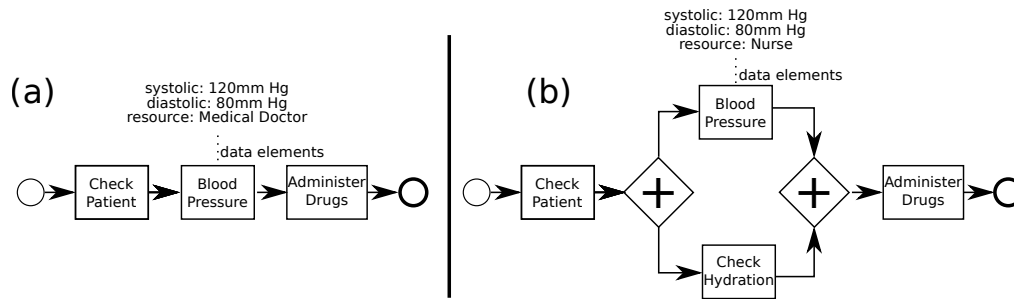


Figure 5.20: Concept drift resulting in adapted process model – medical example

A time sequence is defined as follows in [GRM15, p. 208]: *A sequence of time-stamped data for which the attribute values are the result of measurements of a quantitative real-valued state variable, denoted by $y \in \mathbb{R}$, $y = (y(t_1), y(t_2), \dots, y(t_n))$.*

The challenge is to compare the time sequences in order to detect differences in the associated sensor event streams that can lead to drifts. To compare two time sequences, an alignment is calculated to determine the distances from one sequence to another. The most common distances measure are the Euclidean Distance (ED) [FRM94] and Dynamic Time Warping (DTW) [BC94]. While ED has several advantages like linear computing time and being straightforward, it requires time sequences to be of the same length and is deceptive for noise. DTW is also able to globally find the best alignment and can cope with sequences of different length. The complexity is quadratic, since a $m \times n$ matrix has to be constructed, where m and n are the lengths of the time sequence.

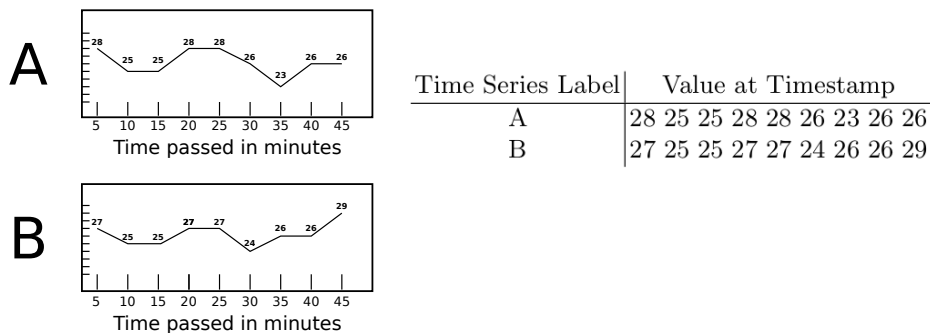


Figure 5.21: Plot of two time sequences and their corresponding values

29	18	12	12	9	9	8	11	8	8
26	17	8	8	9	10	5	8	5	5
26	15	7	7	8	9	5	8	5	5
24	13	6	6	7	7	5	5	7	9
27	9	5	5	3	3	4	8	9	10
27	8	3	3	2	3	4	8	9	10
25	7	1	1	4	7	8	10	11	12
25	4	1	1	4	7	8	10	11	12
27	1	3	5	6	7	8	12	13	14
B	A	28	25	25	28	28	26	23	26

Figure 5.22: Warp matrix constructed using DTW: orange cell = distance

DTW is used, since it is able to deal with sequences of different lengths. Figure 5.21 shows time sequences A and B, together with a table containing the exact values at every timestamp. The $m \times n$ matrix D for the alignment between A and B is constructed by starting in the bottom row and filling every value from left to right, as can be seen in Fig. 5.22. The distance as absolute difference between the actual values is calculated, so in the bottom left corner it is $|28 - 27| = 1$. Afterwards the cheapest cost from one of the cells before is used, so $D(n, m_1)$, $D(n_1, m_1)$ and $D(n_1, m)$ is added to the distance and assigned to $D(n, m)$. The definition follows [SC07]: $D(i, j) = \text{Dist}(i, j) + \min[D(i - 1, j), D(i, j - 1), D(i - 1, j - 1)]$

The distance is found in the top right corner of Fig. 5.22. The alignment can be found using back-tracing starting in the top right corner and following the path back to the start cell in the bottom left corner, i.e., the green cells.

This sections employs the DTW Barycenter Averaging (DBA) [PKG11] algorithm. DBA uses DTW as distance measure and calculates the average time sequence for a set of time sequences. It starts by an arbitrary average sequence and adapts it iteratively by trying to minimize the sum of squared DTW distances from the average sequence to the set of sequences. The computation time of this technique is again quadratic, since a DTW matrix has to be created for each iteration.

5.3.2 Time Sequence Assignment and Root Cause Detection

This section details the components of this approach, i.e., how to utilize time sequence data from sensor event streams to flag process instances for closer inspection when performing a root cause analysis for concept drifts. Note that an analysis for both cases is possible, i.e., finding reasons for concept drifts that have already been detected (ex post) and – particularly during runtime – detecting and analyzing deviations in the sensor event streams that might lead to a future concept drift, i.e., a process evolution.

The architecture of the solution (cf. Fig. 5.23) is used as foundation for the subsequent considerations. Note that the components *Time Sequence Module* and *Drift Decision Detection* (both red) realize the contribution.

For detecting drifts, sensor event streams are taken as input. They can be fed into the system by any process execution engine. In the manufacturing scenario presented in this section, the Cloud Process Execution Engine CPEE⁸ is utilized. The sensor components

⁸<http://cpee.org/>

provide data streams collected through tasks in ⑤ (Fig. 5.26). The process history is therefore extended to include all the data from the sensors. Further implementation details will be described in Sect. 5.3.3.

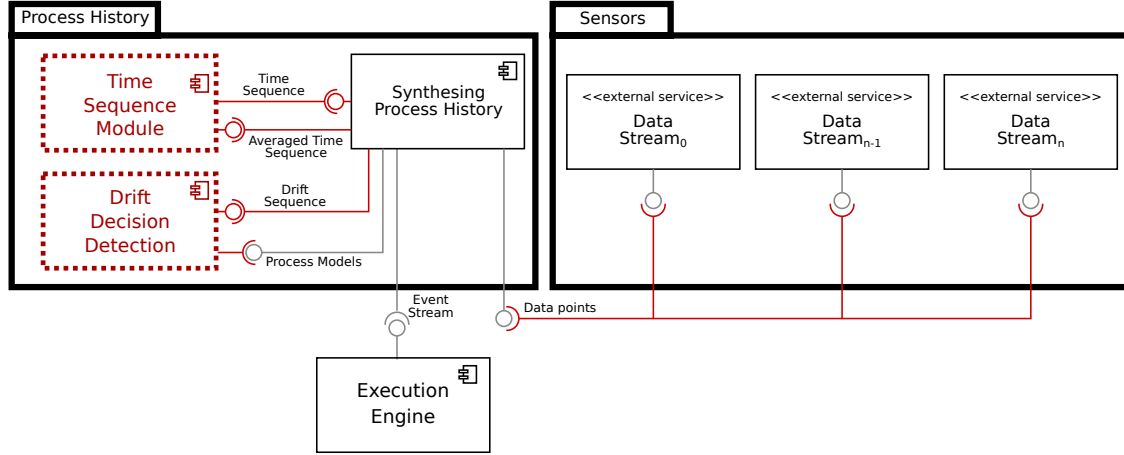


Figure 5.23: Proposed architecture: red parts denote the contribution of this section

Time Sequence Module

This component enriches the process history by adding the average time sequence of every sensor to each new viable process model M_n . To relate a time sequence of an event stream produced by a sensor to a specific process instance, the timestamps of the first and currently last event of the stream are taken into account and the corresponding time sequence is cut out for the process instance. In the example (cf. Fig. 5.20), time sequences between the start time of “Check patient” and end time “Administer Drugs” are mapped to a process instance for both sensors, temperature and humidity.

Algorithm 8 shows the pseudo code for the *Time Sequence Module*. The set of unfitting traces T is provided by the process history, i.e., those traces that do not conform to the current model M_n . The time sequences are provided by external sensors through the process history. In line 8, the return value is initialized as an empty dictionary. A dictionary here reflects a hash table [CLRS09] data structure with a key and a related value to it. Line 8 starts the iteration over time sequences of each sensor. At first, the time sequence for each trace out of T is collected starting in line 8. A time sequence is mapped from a sensor to a trace by beginning from the first time stamp of this trace to the last known time stamp of this trace, as can be seen in line 8. Since an online environment is used, it is possible that traces just started and contain only one event, which results in no time sequence for this specific trace.

Another important aspect of the online setting is, that each trace could have greatly varying execution times, since it is not known how long a complete trace is going to take. To diminish the impact of outliers and faulty or aborted instances, sequences with a duration shorter than the first quartile minus 1.5 times the IQR (Interquartile Range) or with a duration greater than the third quartile plus 1.5 times the IQR, are excluded. The IQR is calculated here between third and first quartile. Other methods for detecting outliers can be applied here or even working with every trace.

The quartile are calculated at (lines 8 and the IQR at line 8). Afterwards the outliers of the collected time sequences are removed. Otherwise the time sequence will be taken into account (lines 8- 8). In the last step (line 8), the averaged time sequence is put into the dictionary *ATS* with its corresponding sensor id as key. The dictionary of averaged time sequences is then sent back to the process history. The current viable process model in the process history is thereby extended by this dictionary, which is then used by the **Drift Decision Detection** component.

Input: *ST*: dictionary of a time sequence for each sensor ID

Result: *ATS*: dictionary of an averaged time sequence for each sensor ID

```

ATS = dict()
for w, ts in ST do
    // w is id of sensor, ts its corresponding time sequence
    temp_ts_list = list()
    stats = list()
    for t in ts do
        if  $|t| < 2$  then
            | next
        end
        temp_ts_list.append(time_sequence(t.first_event.timestamp, t.last_event.timestamp))
    stats.append(temp_ts_list.last.length)
end
    first, second, third, fourth = quartile(stats)
    x = iqr(stats)
    ts_list = list()
    for t in temp_ts_list do
        if  $|t| < first - x \parallel |t| > third + x$  then
            | next
        end
        ts_list.append(t)
    end
    ATS[w] = dba(ts_list)
end
return ATS

```

Algorithm 8: Find relevant time sequences and compute avg. time sequence

Figure 5.24 shows an example of Alg. 8 where the following 5 sequences for the room temperature have been collected: (27, 29), (27, 27, 28, 27, 29, 27, 29, 28], [30, 30, 30, 30, 29, 27), (27, 30, 29, 30, 29, 27), (30, 27, 29, 29, 28, 28, 30, 29, 30, 29, 30, 29, 28, 29). The third quartile for the lengths of these sequences would be 8, the first quartile is 6. Therefore the IQR equals 2. This excludes sequences which are shorter than 3 or longer than 11. The first sequence (27, 29) and the last sequence (30, 27, 29, 29, 28, 28, 30, 29, 30, 29, 30, 29, 28, 29) are not taken into account for the calculation and are printed in a green dashed line. The red time sequence shows the calculated average time sequence with DBA.

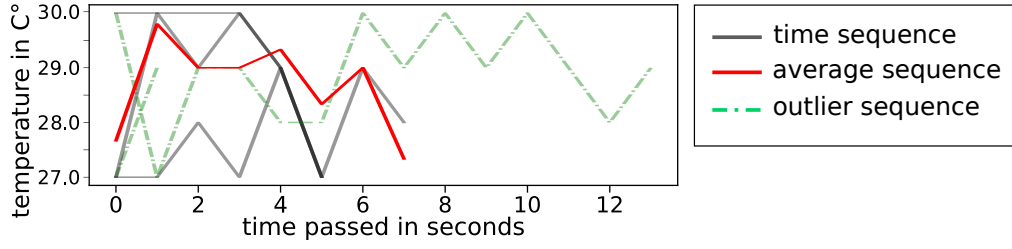


Figure 5.24: Exemplary Result of Alg. 8 The red line is the average sequence calculated using DBA. The green dashed lines represent outliers, potentially due to a faulty process instance.

Drift Decision Detection

The second component of the solution proposed in this work is the **Drift Decision Detection** component.

Input: M : A list of process models with averaged time sequence dictionary

TH : Dictionary of thresholds for similarity

Result: D : Set of sensor IDs that are likely to have caused a drift

$D = \text{set}()$

```

for  $ws\_id$  in  $M.second.ATS.keys$  do
  if  $ws\_id$  not in  $M.first.ATS.key$  then
    | next
  end
   $ts\_a = M.first.ATS[ws\_id]$ 
   $ts\_b = M.last.ATS[ws\_id]$ 
  if  $dtw(ts\_a, ts\_b) > TH[ws\_id]$  then
    |  $D.add(ws\_id)$ 
  end
end
return  $D$ 

```

Algorithm 9: Detecting a set of sensor data streams which caused a drift

Algorithm 9 shows the detection of the most likely external sensors that can have caused the drift in the process model. The process history sends two process models to this component in order to receive a set of external sensors which caused a drift from the first model to the second model. Each of these process models contains its average time sequence for each external sensor. The dictionary TH is user defined and contains for every external sensor, a related threshold for the distance between the two average time sequences. A different threshold for each external sensor is needed, because the dynamic warp distance is calculated using the differences in the data points. Assume that for the running example (cf. Fig. 5.20), the ideal temperature ranges between 27° and 29°C. Thus similar time sequences have a low absolute cost depending on the length of the alignment. A sensor keeping track of parts with higher tolerances can therefore have a higher warping distance for similar sequences. These thresholds can be approximated using a test set for the classification, where an expert has to define sensitivity and specificity for the sensor data. At the start, the return value D is initialized as an empty set in line 9. The loop

iterates over every key that is in the dictionary of averaged time sequences (ATS) in the second model in line 9. It is to be noted that the models could have different events attached to them, but the external sensors should be the same. If a specific sensor is not present in both models, it cannot be taken into account, see lines 9. The average time sequence for one sensor is retrieved for both models in line 9 and 9. If the cost of the alignment, which is reflected in the top right cell of the warping matrix, see Fig. 5.22, is greater than the corresponding threshold to the sensor, this sensor is added to the return value D in line 9. This set of external sensors is then returned to the process history, where it is stored.

Performance Optimizations

One problem with DTW is the computation time, since a $m \times n$ matrix has to be constructed, where m is the length of one time sequence and n the length of the other time sequence.

One approach that can be used to optimize the performance of Alg. 8 and Alg. 9 is FastDTW [SC07]. FastDTW aims at performing DTW in linear time with 3 steps. First the time sequence is shrunk into smaller time sequences that reflect the same curve approximately. Then the minimum distance warp path is computed for the smaller time sequence. Afterwards this warp path is adjusted to the original time sequence. For a length of 10000 data points the computation time can be reduced from 57.45 seconds to 8.42 seconds. The error rate for this approximation is below 1%.

Another way to speed up time warping is early abandoning [WKVHMN05, JY09]. In this strategy, if the warping distance is above a certain threshold while creating the warping matrix, the algorithm can stop the execution and label it as an outlier.

Both methods are suitable optimizations for Alg. 9, since it uses user defined thresholds for each external sensor, but not for Alg. 8. This is because the average time sequence is computed using the complete DTW distance score, thus not exact methods like FastDTW and early abandoning cannot be used. In Sect. 5.3.3, Alg. 9 is evaluated using DTW and FastDTW.

5.3.3 Evaluation of Drift Explanation Discovery using Real-World Data

The algorithms and components presented in Sect. 5.3.2 are prototypically implemented and tested based on a real-world IoT application from the manufacturing domain in order to prove the effectiveness and feasibility of the approach: The Austrian Center for Digital Production⁹ produces parts called GV12 for a gas-turbine (see Fig. 5.25) as a prototypical solution for a customer. The requirements for the part include high precision manufacturing (low tolerances, i.e. some aspects allow for deviations of only 0.02 mm), and strict quality assurance for each part, including (a) detailed tracking of manufacturing data for each part and (b) measuring the adherence to tolerances for more than 12 features with automated precision measurement equipment.

The entire production is carried out automatically by implementing the interaction between the involved machines through industrial robots and transport systems. Here, the focus is on the manufacturing and quality control as shown in Fig. 5.26. Currently more than 20 processes and sub-processes are involved, and orchestrated during production of up to 40 parts per batch. Figure 5.26 illustrates the basic manufacturing logic:

⁹<https://www.acdp.at/>

- ① Batches of up to 40 pieces are ordered, the manufacturing is scheduled.
- ② The interaction between all machines and robots is orchestrated, while enforcing industrial safety principles¹⁰.
- ③ Individual parts are produced by using the following three steps:
 - Machining of a part from hardened steel, which takes about 4 minutes per part.
 - Measuring of the part by a high-speed optical micrometer¹¹, while the next part is machined. This takes about 12 seconds per part.
 - Measuring of a part with automated precision measurement equipment¹², which takes about 8 minutes per part, and is also done in parallel to the machining.
- ④ A generic machine monitoring process determines when to start data collection for both, machining and measuring.
- ⑤ A generic data collection process produces a continuous stream of values when the laser of the high-speed optical micrometer is scanning the surface of the part.

The “Measure with Keyence” task is done automatically by a Keyence measuring machine at no additional cost in parallel to the production of the next part.



Figure 5.25: GV12 part

As the Keyence machine is very compact, fast, and operates without touching the part, this step is done after the robot extracted the part from the production machine, and before it puts it on the palett. On the palett it is transported to the MicroVu measuring machine, which is rather big and has to be operated in a location with low vibrations and special light and temperature conditions. The task “Measure with MicroVu”, as opposed to the task “Measure with Keyence”, is required by the customer, because it basically creates an objective report about the quality of a part.

After some time, deviations in the process event stream collected by ⑤ can be observed. These deviations can happen based on

- physical effects due to deteriorating machining tools, or temperature fluctuations.

¹⁰<https://www.iso.org/standard/51330.html>

¹¹<https://www.keyence.com/products/measure/micrometer/ls-9000/index.jsp>

¹²<https://www.microvu.com/products/vertex.html>

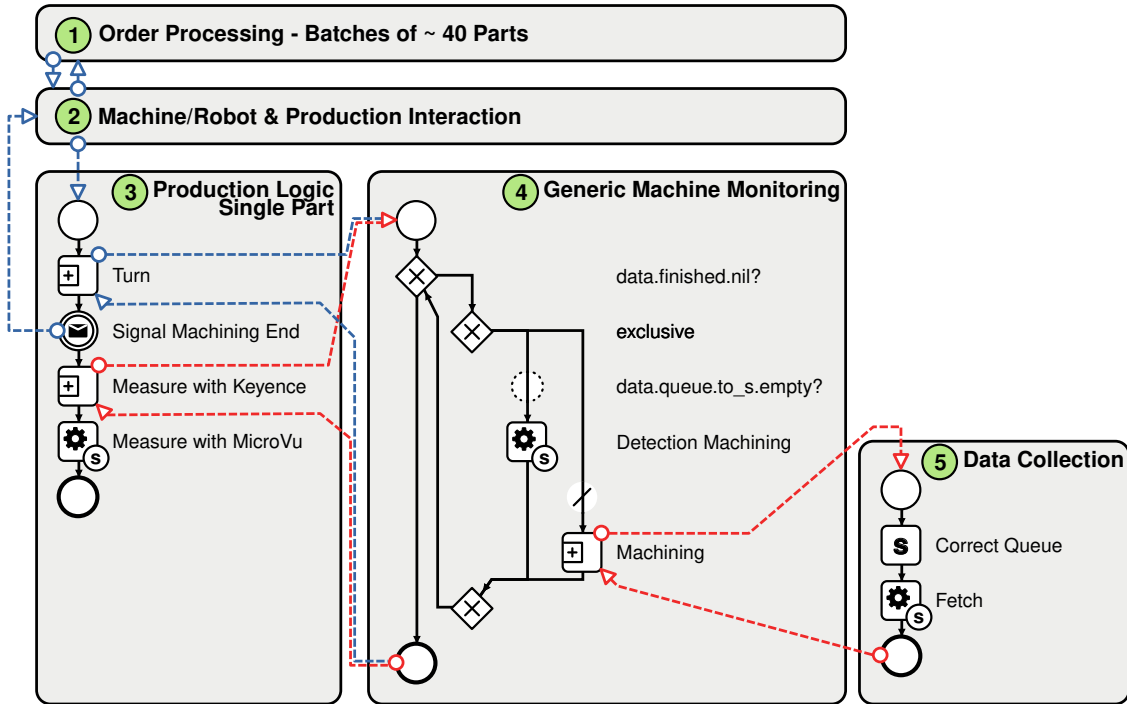


Figure 5.26: Batches of GV12 parts

- problems stemming from accumulating debris that affects the production quality as well as measurement quality.

Up to this point only the extreme values of the time sequence (i.e. min, max) from the Keyence machine were used, which are sufficient for detecting (a) if the part has been dropped by the robot (no part), or (b) the part appears to be too big (i.e., it is engulfed by chips). However, the extreme values proved not to be effective for early detection of parts which do not comply to the quality requirements. If an early reliable estimation of the quality was available, it could be used to skip the ‘Measure with MicroVu’ altogether, which would save valuable resources.

Hence, our approach for this evaluation is instead of taking only the extreme values of the measurement into account, to analyze the complete time sequence of the measurement. Every time the process history detects a drift in the data elements, i.e., “Measure with MicroVu” detects only faulty instances, a drift has been detected in the data model. Algorithm 8 calculates then the average sequence, e.g., Fig. 5.27. The threshold for Alg. 9 is here calculated ex-post, with the results of “Measure with MicroVu”.

Prototypical Implementation

The orchestration of the BPMN 2.x based process models on the factory floor (cf. Fig. 5.26) is driven by the process engine CPEE¹. The process history component subscribes to the CPEE in order to receive information about every executing event. The external sensor represented by activity “Measure with MicroVu” is a high-speed optical micrometer¹³.

¹³<https://www.keyence.com/products/measure/micrometer/ls-9000/index.jsp>

The data set¹⁴ contains 1026 traces in the XES¹⁵ format for 37 parts and is available at the figshare repository [SRMM]. The traces are produced by 13 different process models. The sensor values amount for 6.2 MiB out of 1 GiB of total data. A time sequence for this event contains on average about 776 data points. Measurements from a time sequence range from 4.09 up to 37.87 millimeters.

The process history creates the process models as described in [SRM18]. The process history uses a sliding window approach to deal with infinite amount of data that is being captured by listening to streams, i.e., only a specified number of traces are used for detecting a new process model in the history.

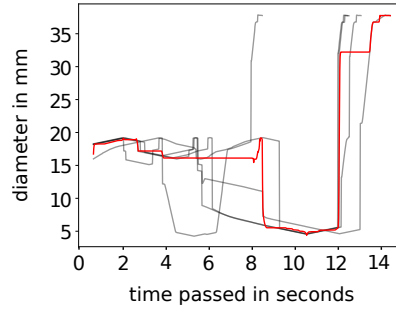


Figure 5.27: Result of implementation. The red line represents the average sequence

The **Time Series Module** component is implemented in Python as the **tslearn** package [Tav18] is used, because it provides functions for computing an alignment using DTW as well as DBA for finding the average time series. The results of this component are retrievable via a RESTful web service as well. The **Drift Decision Detection** component also uses these libraries. The result of Alg. 8 on the data set is depicted in Fig. 5.27. Each grey sequence relates to one specific trace and shows the measurement data points of one part. As it can be seen, one time sequence only lasts for about 8 seconds, while the other ones last about 12 to 14 seconds. The average sequence, calculated using DBA is depicted in red. This sequence is stored as additional information in the process history for the current process model. Since this log only provides one sensor, i.e., “Keyence”, only one sequence has to be calculated using DBA. To determine the feasibility of the implementation, the following questions is looked at:

- How are parameters such as the duration of a process instance or the number of traces in the process history sliding window, affecting the algorithms?
- What is the performance of these algorithms?

The performance of Alg. 9 is only depending on the length of the average time sequence, which is hard to tweak with parameters of the process history and the number of sensors. It can be adapted by changing the amount of data points that are to be stored per time unit. The performance of Alg. 8 on the other hand is highly dependent of the parameters set for the process history.

In order to rate the effectiveness of the approach it is possible to rely on the data provided by “Measure with MicroVu”. Out of 37 parts, 18 parts were faulty. With this

¹⁴<http://gruppe.wst.univie.ac.at/data/timesequence.zip>

¹⁵<http://xes-standard.org/>

Table 5.7: Results of both Algorithms

Window Size	False Positives	False Negatives	Runtime
1	45%	0%	1.4s
5	0%	0%	10.3s
10	0%	0%	20.7s

knowledge the threshold is varied in order to achieve 0% false negative detection of parts. In other words: no parts that are faulty should be delivered to the customer, on the other hand it is acceptable that some parts that are actually good are detected as faulty. The optimal threshold proved to be 22.

When varying the window size, i.e., the number of traces to be analyzed during the drift detection, the following results emerge: As can be seen in Tab. 5.7, a window size of 5 and a threshold of 22 is sufficient for our scenario. With these values 100% of the faulty parts can be identified, without relying on the time intensive “Measure with MicroVu” task. This means that for a rate of 18 faulty tasks, almost 50% of the production time can be saved, based on calculation of drift for sensor event streams.

Concept Drift Prediction / Process Evolution

For the task “Measure with Keyence” in Fig. 5.26, as collected by process (5), the deviations for the measurements between parts have some serious repercussions that can lead to multiple possible process evolutions.

The results were discussed with three domain experts involved in the production of the GV12 parts. When discussing the results from the drift analysis, the domain experts came up with the following discussion points.

As can be seen in Fig. 5.28, the machining produced long chips, which entangled the part. Furthermore, the comparison of the drift for “Measure with Keyence” with the quality data from “Measure with MicroVu” (see Fig. 5.26) was deemed sufficient for predicting the quality of a part, thus allowing for immediate removal of faulty parts from production which decreases the overall time per batch greatly: the less “Measure with MicroVu” the better. This led to the proposal of the concept drifts / process evolutions shown in Fig. 5.29. Overall, the concept drifts can be classified as follows:

- Static Evolution (a): an extra activity “Chip Removal” was proposed to be inserted, based on the observed drifts. A robot blows compressed air on the part, to remove



Figure 5.28: Chips on GV12 - wrong measurement

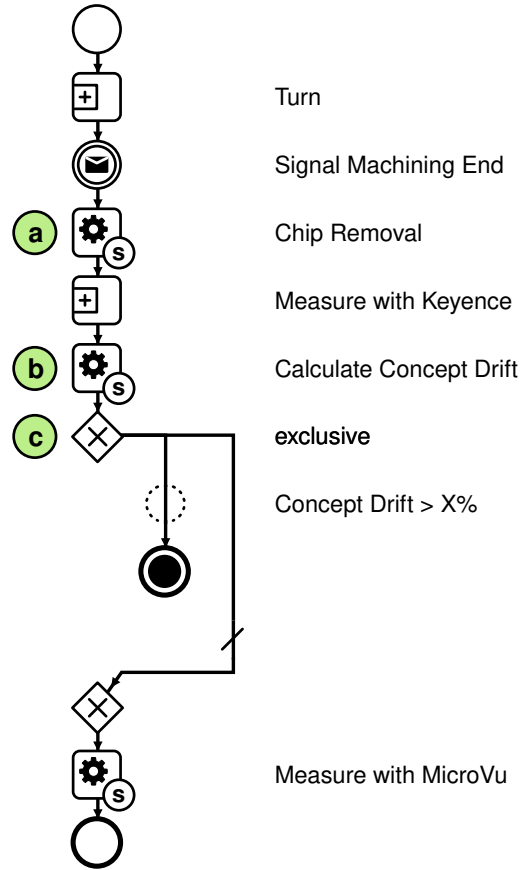


Figure 5.29: GV12 Prototype Part

debris and chips, which allows for more accurate measuring. This will allow for lower possible thresholds in future / similar scenarios.

- Dynamic Evolution (b)+(c): The drift is to be actively calculated at runtime, based on previous process instances, and made available to the current instance. A decision (c) is proposed to be inserted, that allows for terminating single parts without “Measure with MicroVu”.

Performance evaluation: Table 5.8 shows the runtime of Alg. 8 to analyze the applicability of this solution. Random time sequences are generated with 10, 100, and 1000 data points on average. Algorithm 8 is then applied on a set consisting of 5, 10, 50, and 100 time sequences. The results of 10 data points on average show, that the execution time of Alg. 8 for 100 sequences is even lower as the one for 50 sequences. This happens, because the time for the calculation is so small, that other currently running tasks of the operating system may interfere with the execution.

With 100 data points, Alg. 8 affects the total execution time to a greater extent, especially with more sequences: 50 sequences result in a more than 3 times longer execution time than 10 sequences. With 1000 data points, the execution time with more than 50 sequences is increased by more than 10 times the execution time with 10 sequences.

Table 5.9 shows the comparison between DTW and FastDTW (cf. Sect. 5.3.2) in

Table 5.8: Runtime of Alg.8.

Datapoints/ Sequences	10	100	1000
5	1.17s	1.22s	3.53s
10	1.18s	1.31s	7.01s
50	1.29s	4.43s	75.83s
100	1.22s	4.63s	133.00s

Table 5.9: Runtime of Alg.9.

Datapoints	DTW	FastDTW
10	0.88s	0.0002s
100	0.87s	0.0003s
1000	0.91s	0.001s
10000	4.70s	0.01s

terms of speed. As expected, FastDTW is the faster technique as it works in linear time. Unfortunately the results differ greatly for DTW when compared to FastDTW. While, for example, the distance between 2 sequences with random values between 90 and 110 and 10000 data points was 343.2 when using DTW, the distance equals to 45299 when using FastDTW. Since both algorithms are highly depending on the global maximum of the alignment of sequences, FastDTW is not a suitable option.

Assessment by domain experts: The results are presented to a machine operator, a mechanical engineer, and a measurement engineer. All three experts were overall satisfied with the results. They highlighted that to the best of their knowledge in order to achieve similar results, additional – hard to configure – software would be necessary.

5.3.4 Discussion of Drift Explanation Discovery

Possible **limitations** in the context of the presented approach include:

- *Performance:* An important aspect for the performance of this approach is the number of data points in a time sequence. As can be seen in Tab. 5.8, even with 1000 data points and 10 time sequences the implementation took about 7 seconds. This of course increases linearly with number of sensors. Other techniques like FastDTW instead of DTW, reduce the runtime drastically, but the alignment using FastDTW varies greatly from the globally best alignment using DTW, which leads to worse results.
- *Sensors selection:* While in general IoT devices such as external sensors provide a valuable source for detecting the cause of a concept drift, choosing the “right” IoT device may be hard in some cases. The reason is that in many real-world scenarios there is a plethora of devices creating data streams and therefore time sequences. Taking an external sensor into account, that has no relation to the process model, for example, can produce wrong results, since the time sequences of this sensor may vary to a great extent and hence be incorrectly identified as the source of a drift. In addition, the runtime is heavily depending on the number of sensors, hence not significant sensors should be excluded. Therefore it is recommended that an expert additionally validates the results. If no sensors can be

excluded by experts, a parallel optimization is advised of Alg. 8 where each sensor can be calculated separately. This reduces the execution time of the algorithm to the execution time of the sensor with the most data points.

- *Thresholds*: Another important aspect is finding the threshold for Alg. 9 automatically. If there is a training set, the threshold can be calculated until a specified sensitivity and specificity are met. Otherwise, an expert sets the threshold.

Also, the following **threats to validity** have to be considered: The data set of the evaluation comprises the data of one sensor. Hence, the selection of the sensors cannot be evaluated. While the increase of the runtime is predictable, the quality of the results can differ greatly, if not related sensors are taken into account. The real-world case comes from the manufacturing domain, where the selection of the sensors may be easier, since the conditions of the events are often in a controlled environment, like a factory. In other domains, like the medical domain or logistic domain where numerous external data stream sources can affect the execution of a process, the selection may be more difficult. In future work, experiments in different domains are planned.

5.4 Conclusion and Outlook

This chapter focuses on the execution of a process and analyzes the behavior of process instances to a process model.

An advanced cost function to calculate the deviation cost in an alignment for conformance checking in an online and offline way is introduced as well. The advanced cost function allows conformance checking algorithms to alter the deviation costs for an alignment. The data elements that are attached to events are used to distinguish errors in the logging and security breaches, giving a more detailed view on the results of standard conformance checking. The required data to make this distinction is gathered from an process execution log. The results are promising.

Temporal conformance checking is introduced, in order to detect and quantify temporal deviations for task durations and temporal distances between events. The approach can be executed in an offline and online setting. For an offline setting, a process execution log can be split in training and test sets. Based on the training set and the process model, a temporal profile is calculated. In addition, the tasks in the process model can be annotated with their significance for temporal conformance. The temporal profile and the model can then be compared to the test set (offline) or an even stream of interest (online). A drawback of this approach is the requirement of lifecycles attached to events, in order to distinguish start and end events. Without this, the distance within events cannot be detected at all and the distance between events can only be guessed.

A novel approach to predict the root cause of a concept drift in a business process based on external sensor streams is presented, consisting of two algorithms to compare time sequences associated with sensor event streams in combination with a process event stream. The algorithms are capable of detecting the drifts in the external sensor event stream with high accuracy.

To summarize, the main contributions of this chapter are:

- Conformance checking usually focuses on the workflow perspective of a process instance and uses a trivial cost function for deviations in an event sequence.. The approach introduced in Section 5.1 extends the conformance checking algorithms

by defining an advanced cost function for deviations and analyzes deviations on the data perspective as well as the workflow perspective.

- Temporal deviations are analyzed in Section 5.2. The section enriches process models by a temporal perspective, i.e., the temporal distance between events and the duration of events. In addition a new algorithm is presented to quantify the cost of temporal deviations.
- Section 5.3 investigates the source of a concept drift of a process. The presented approach aligns data points, produced by external sensors as a data streams, to a process instance and process model. Each time a concept drift is detected, the data points for a process instance are interpreted as a time sequence and compared to a time sequence of the process model, to explain why this drift happened.

As future work, a personalized generation of temporal profiles is planned, i.e., young patients could need a shorter recovery time for the process depicted in Figure 5.3 than older patients, thus the temporal profile should feature a different temporal distance between the “Cholecystectomy” and “Hospital Dismissal”. Further predicting and explaining of future drifts as well as analyzing hierarchical data elements, like an organizational chart, using the advanced cost function is planned. The next chapter, Chapter 6, the requirements for the algorithms presented in this thesis, are evaluated using a focus group study and a technical action research study.

6 Evaluating TIDATE - Time and Data Aware Process Mining at Runtime

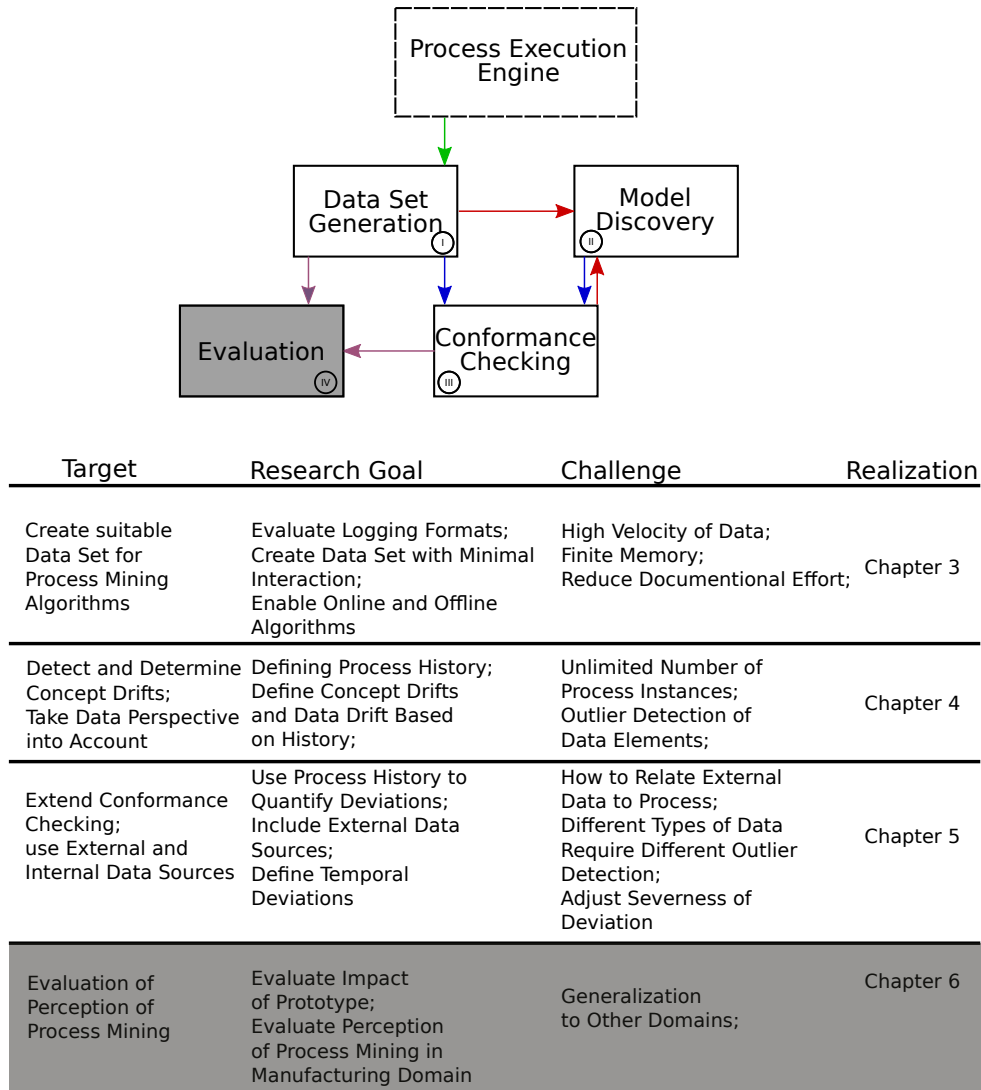


Figure 6.1: Overview of Targets, Research Goals, Problems of this thesis. The features for this chapter are marked.

This chapter addresses objective ④ in Fig. 6.1, i.e., the evaluation of the framework of this thesis, i.e., the algorithms presented in the previous chapters, the generation of a XES event stream (cf. Chapter 3), the discovery of the process models based on an event

stream (cf. Chapter 4) and conformance checking based on different areas (cf. Chapter 5).

To evaluate the collection of the algorithms of this thesis as an artifact, a technical action research (TAR) [WM12] study has been chosen, since it focuses on an artifact which is developed closely with potential stakeholders. In addition a focus group study [Kru14] has been conducted. This study allows to identify the requirements, the desired outcome as well as the actual results of the framework. Through both studies, the following research questions can be answered:

- RQ ④ What are the general expectations on process mining algorithms by domain experts and what are the actual results after it has been introduced?

Section 6.1 provides the evaluation of the TIDATE framework using TAR as research method. The algorithms are applied on an event stream generated by a process execution engine, CPEE [MRM14] (cf. Chapter 3), where deviations are detected. The discovery of process models is not evaluated as part as the TIDATE framework, since the process models have been designed already by domain experts and a process execution engine is already in use. In Section 6.2, the focus group study is presented using a double-layer design involving two different companies and employees on three different hierarchy levels in their organization.

A selection of text, figures and tables within this chapter is based on the following publications.

Stertz, F., Mangler J., Rinderle-Ma S.: The Role of Time and Data: Online Conformance Checking in the Manufacturing Domain. Tech. rep. (2021), <https://arxiv.org/pdf/2105.01454>

Stertz, F., Mangler J., Scheibel B., Rinderle-Ma S.: Expectations vs. Experiences–Process Mining in Small and Medium Sized Manufacturing Companies
In: International Conference on Business Process Management Forum, Pages: 195-211
https://www.doi.org/10.1007/978-3-030-85440-9_12

6.1 TIDATE - Artifact Design

While the previous chapters, introduced new algorithms to extend the current established process mining algorithms, the design of a framework consisting of these algorithms and a possible implementation needs to be looked at. From a methodological point, Technical Action Research [WM12, Wie14] is employed. *“Technical action research (TAR) is the use of an experimental artifact to help a client and to learn about its effects in practice”* [Wie14]. Doing so, TAR constitutes the last step *“from the conditions of the laboratory to the unprotected conditions of practice”*. This study focuses on the framework of this thesis, TIDATE as experimental artifact. TIDATE consists of a lightweight and modular process engine for modeling and executing the manufacturing processes and the presented algorithms in the previous chapters.

The findings are promising. From the different process mining tasks, process discovery seems to play a minor role in manufacturing whereas the importance of online conformance

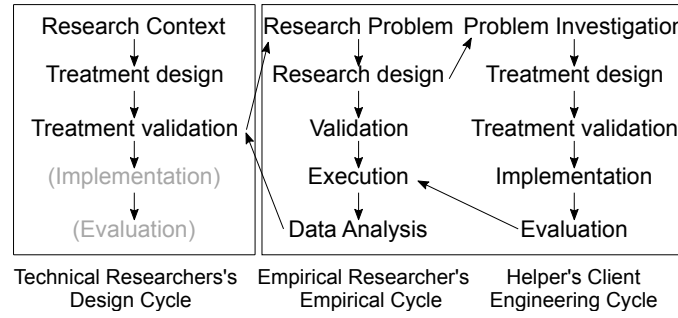


Figure 6.2: TAR cycles (based on [Wie14])

checking is emphasized. The role of time and data in conformance checking is emphasized, too, i.e., shopfloor workers and process supervisors are actually supported in their process monitoring task. The generation of data sets that are suitable for a product-oriented analysis is possible and does not conflict with the batch-oriented modeling of the processes. Linking sensor data requires a certain upfront effort.

The section is structured as follows. The methodology and setup of the study is explained in Sec. 6.1.1. The implemented approach is then tested and evaluated in Sec. 6.1.4. Section 6.1.5 answers the knowledge questions of the approach and discusses further improvements.

6.1.1 Methodology

This study employs Technical Action Research (TAR) [WM12, Wie14] aiming at the validation of an artifact in a realistic case, i.e., the implementation in a real-world organization. This is a major distinction to other forms of Action Research as TAR is technology-driven instead of problem-driven. Section 6.1.1 outlines the research context of this study, Section 6.1.2 the research design, and Section 6.1.3 the artifact validation.

This section provides an overview of the artifact, applied in a real-world application and on the research method used to validate the artifact. This artifact is designed and then evaluated using technical action research in the following section 6.1.4.

Research Context

In TAR, the researcher takes three logically separate roles, i.e., **technical researcher**, **empirical researcher**, and **helper**. Consequently, the researcher in TAR, is allowed to guide domain experts using the artifact, to interfere, and aims to answer research questions with the results of the experiment.

Figure 6.2 (based on [WM12]) shows the three cycles used in TAR that reflect the three roles of the researcher in TAR.

The technical researcher's design cycle defines the research context of a study. According to [Wie14], at first, the research problem is investigated and the **research context** is defined by describing the goals of the created artifact and the current knowledge of the environment. For this study, the experimental artifact is the lightweight process execution and mining framework TIDATE and the practical setting is a manufacturing environment. The clients are the domain experts, i.e., the shopfloor workers and the process supervisors.

In this study, the clients can benefit from time and data-aware conformance checking techniques by detecting errors in the behavior of the manufacturing processes as soon as possible. Moreover, the clients benefit from a minimally invasive generation of product-oriented logs/event streams based on their batch-oriented process models. In the following, the research context is defined by dividing the goals into knowledge and improvement goals.

Knowledge goal: Can TIDATE help shopfloor workers and process supervisors in a manufacturing environment in a useful way?

Improvement goals:

- Generate event streams for time and data-aware conformance checking during runtime in a product-oriented manner.
- Highlight time deviations in process instances as soon as possible.
- Highlight data deviations linked to process instances as soon as possible.
- Highlight deviating behavior in the execution order of events in process instances as soon as possible.

Current Knowledge: Process mining techniques have been evaluated using real-world process execution logs. However, these evaluations are usually still based on laboratory settings, i.e., the techniques have not yet been applied in a live real-world setting.

The **treatment design** yields TIDATE as a lightweight process execution and mining framework to achieve the previously defined goals. For **treatment validation** TIDATE is validated performing the **Empirical** cycle of the TAR study. In the protocol for the TAR study the research problem is defined as follows using the checklist from [Wie14].

Conceptual Framework: TIDATE enables the logging of process instance executions and the detection of deviating behavior in the execution.

Knowledge Questions:

- How can domain experts easily use process mining techniques?
- Can the results provided by process mining techniques be used by domain experts?

Population: The population for this approach consists of companies which already have knowledge of their processes and focus on the correct execution of process instances.

The validation of TIDATE and the answer for the research problem are generated by implementing the treatment at a potential client.

For the client selection process, a suitable client was identified through previous collaborations in different projects. The resources of the client are reachable through web services. Hence only small adaptations of TIDATE were necessary to deploy it at the client's site. There are threats to generalizability, since the research applied the elaborated approach, but domain experts have been shown the results and developed process models inside the process execution engine to witness the usability of the approach.

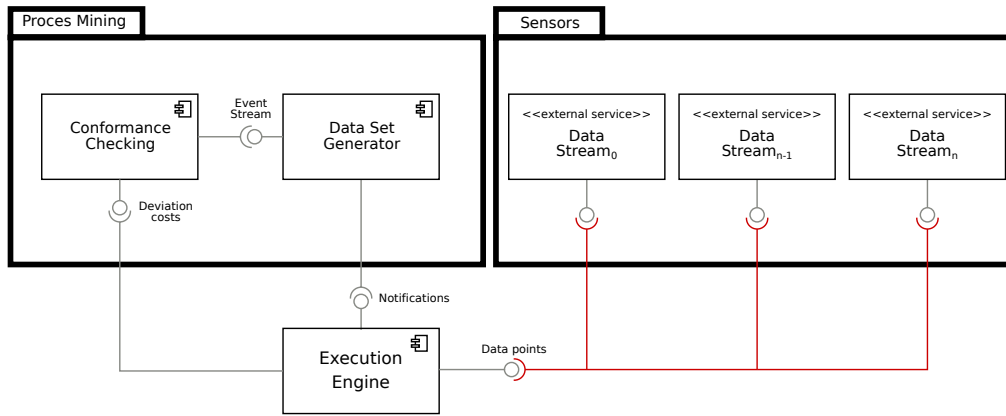


Figure 6.3: TIDATE architecture

6.1.2 Research Design

After the research context and research problem are defined and a suitable client is found, the empirical cycle is further defined by the researcher, and the helper's client cycle (cf. Fig. 6.2) is defined by the researcher and the client. Both cycles interact with each other, therefore the coordination starts as soon as the client is acquired and ends after the client cycle is evaluated (cf. [Wie14]). The problem investigation is the starting point of this design and adaptation of TIDATE to the client's needs.

Problem Investigation: The client is interested to know how tasks are being executed, and if a process model is available, to know if the execution of instances are matching the behavior of the related model. Common phenomena for this problem are a different execution behavior due to a missing automatic task enactment like a process execution engine and missing data sets allowing the detection of said behavior.

Artifact Design: The artifact TIDATE, that is introduced and implemented at the client, contains a process execution engine. The client models the manufacturing processes which are then enacted and executed by the process execution engine. Moreover, the engine generates an event stream for the executed process instances. This event stream is then used for conformance checking to give the client feedback. Process discovery is not used, since process models have to be created for the use of the process execution engine. The engine then orchestrates the execution of active tasks. Hence, no unknown process models should be discovered from the event stream.

Fig. 6.3 shows the TIDATE architecture. This architecture can be used as blueprint for (lightweight) process execution and mining frameworks. The process execution engine creates process instances using process models from a repository. The models in the repository are designed by domain experts. The engine then enacts the tasks in the sequence dictated by the model. To generate an event stream and a process execution log, a notification is generated each time a task is enacted, finished, or changed. A description of the process model is also sent as a notification when a process instance is created or the model is changed for the process instance. The Logger component is subscribed to the engine and generates an event stream based on the information received from the notifications, creates an process execution log and stores the description of the related model. This is important for checking the conformance of an instance.

The conformance checking component is receiving the event stream and the model description from the Logger component and computes the alignment costs every time a new event for a trace is detected. Note that the main focus of this framework is on conformance checking in real time. Since conformance checking of the workflow perspective is a time consuming task to detect the alignment with the smallest cost, the framework focuses on conformance of the data elements . The conformance checking component is receiving data from external sensors as well. The process execution engine typically controls the workflow perspective of a process instance, but the data elements attached to events could still contain wrong information. In addition, not every information affecting the execution of a task is shown in an event. External sources, like the temperature in room for example, can affect an instance, as well. To exploit this information in conformance checking, the data from external sensors can be used, which is typically stored as a time sequence. Time sequences can be compared using, for example, dynamic time warping [SRM20b, RCM⁺12].

6.1.3 Artifact Validation

To solve the client's problem of ensuring a correct execution order of the tasks in a process instance and detecting errors as soon as possible, the following data is measured. Domain experts design the process model and together with the researchers create the necessary interfaces for the process execution engine to interact with the machines. After an introduction, the domain experts create and start the execution of new process instances. The event stream is generated by the process execution engine without needed interference from anyone and the behavior is automatically checked on the data perspective of the events. This data involves time sequences, temporal deviations between events and the duration of events as well as other numerical data relating to the configuration of the participating machines. The conformance costs are reported back to the process execution engine where notification events can be produced to inform domain experts. Domain experts are interviewed in the end to see if process instances that have gone wrong are detected correctly with an increased cost, if the information was useful in detecting the exact problem of the process instance as well if problems occurred during the usage of the framework.

In the following Section 6.1.4, the artifact is executed inside the client's environment, results are generated and possible improvements are being created, based on the feedback of the client.

6.1.4 Research Execution

In Section 6.1.1, the methodology and planned actions for applying and executing TIDATE at a client, i.e., a manufacturing environment, were outlined. This section describes the actual research execution along its setup in Section 6.1.4 and its execution and results in Section 6.1.4.

Setup

The process execution engine provided by TIDATE, is used at the client, as it already provides a notification stream to detect process model changes as well as the enactment and

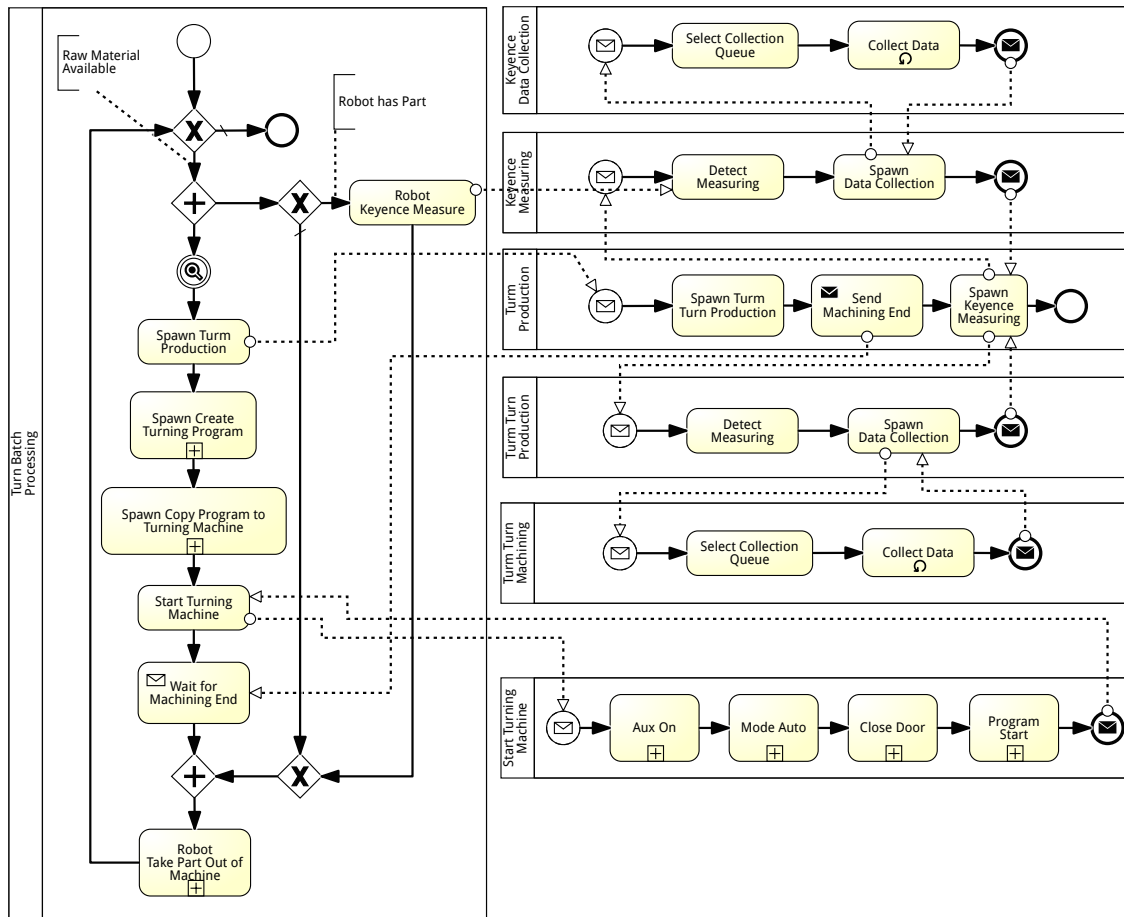


Figure 6.4: Process model used in the process execution engine for producing Turm parts.

completion of tasks. For this application, robotic machines and other software interfaces are orchestrated and controlled by the process execution engine.

A client process driving the production of small metal parts called “Turn” by machines for the usage by another company is depicted in Fig. 6.4. In the beginning the machine’s state is checked and if it is free, the production process is spawned. Note that the whole production process is split into a number of sub processes, so it is easier to read for the domain experts as well as it is easier to maintain small processes. While different parameters are fetched during the “Turn Production” process, the machine is set up accordingly with a program to be deployed on the machine. This allows for a high flexibility to change the process for the machine for each process instance. When the machine is finished, the part is taken out and measured by the Keyence software and afterwards manually by domain experts using MicroVU. In the end it, the produced part is put onto a tray outside of the machine to be used in another process.

The notifications detected by the process execution engine are transformed into an event using the XES format. These events can then be immediately used by the conformance checking algorithms. Additional information which is captured by external sensors is detected by the Fetch task from the “Turm Keyence Measurement” process in Fig. 6.4. The machine is constantly measuring the diameter of the produced parts, to check if the parts

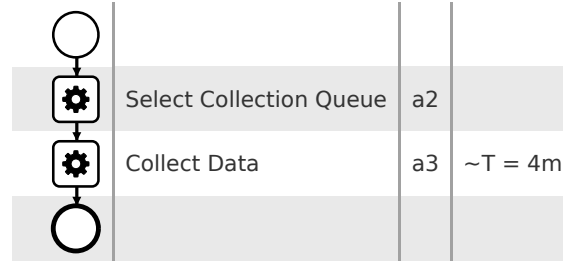


Figure 6.5: Small example of process model in execution engine at client. The time $\sim T$ represents the expected task duration on average.

have been correctly produced. The gathered data points of these sensors are aggregated by the software of the machine, fetched by the process execution engine in the Fetch task and then put into the notification of the task execution. Hence a time sequence for the diameter of parts is generated for every process instance. For detecting imperfect parts, the time sequence of a well produced part is saved additionally in the process execution engine which is compared by using dynamic time warping, see Chapter /refch:cc.

For the detection of temporal deviations, algorithms from Chapter 5 are used. There are 2 types of temporal deviations. The first one is concerning the task duration. If the start and end of a task is supported by the event stream, the task duration can be calculated by determining the difference. The other type of deviation that can occur, is the time distance between the end of a task and the start of the next task. Fig. 6.5 shows a sub-process as implemented in the process execution engine. For task a3, “Collect Data”, a time duration is put into the model. This duration symbolizes the expected task duration on average. If no time is present, a deviation in this task is seen as harmless. The **z-score** [CA86] is used to determine the distance of a witnessed task duration to the task duration in the process model. If the distance exceeds a threshold, typically 3, an outlier, hence a severe deviation, is detected.

To apply process mining techniques with correct results, every produced part should be treated like an process instance. An interesting thing emerged at the beginning of the implementation and setup of the framework at the client. Instead of viewing each produced part as a process instance, the shopfloor workers view the production of a batch as one instance. This means, that if out of a batch of 5 for example, only one instance shows deviating behavior, the conformance score is still quite high, since the other 4 instances show no deviations. For the beginning of this study, the produced parts have been divided hard coded, with a better solution presented in Sect. 6.1.5.

Execution and Results

The process execution engine enacts the different tasks for process instances and generates an event stream. This section focuses on discussing the results of the experiment.

Logging: The client is now able to automatically generate an event stream and even process execution logs for the analysis of executed process instances. This allows for repeatable results using process mining techniques, since the process execution log contains time stamps as well to generate an event stream out of it. The logging component is being executed as a separate process, hence it is never interfering directly with the execution of process instances.

Conformance Checking: Interesting results emerged from the first batch of parts produced out of the data set. By taking a look into the process execution logs of the different sub processes, no deviations are detected concerning the event flow of the process instances, but instance `b20fedd7` of the “IRB2600 Unload to Tray” process, contains time deviations. The “Move up” and “Move down” events, which relate to the tasks `a12` and `a21` with a task in between “wait” relating to `a17`. Compared to other process instances and checked with the domain experts, this measurement, done by external sensors, takes about 30 seconds to move a part through the laser. In the log, all 3 tasks are taking 0 seconds. After further investigation, a software error has been detected, leading to no measurement at all by the external sensors, which yielded a return value immediately, hence the task is logged and detected in the event stream as an event, but with an unexpected time stamp and no values attached to the event. Therefore an error, has been detected as soon as possible and the external software with sensors has been restarted to avoid this mistake in future parts.

Another error that has been detected was due to a collision of the robotic machine arm with another machine. This collision led to a slight misplacement of a mandrel of the machine. The mandrel is still within a certain safety range, hence the robotic arm is still reaching inside the machine. However, the produced part is not at the programmed position due to the misplaced mandrel by a few millimeter. Hence the robotic arm is still allowed to move, but grabs the produced part not correctly. The collision is not detectable in the event stream, since the time stamps, data elements, and event order are correct, but the time sequence from the external sensors during the measurement of the current and following parts yields a different time sequence than the expected time sequence which results in a high distance using dynamic time warping. This is detected by TIDATE. After investigating the collision, the reason has been detected. It has been due to a not logged meddling with the program of the robotic arm, which led to an incorrect execution in the process instance.

Other than the situations described above the process instance of the inspected batch conforms to the process model and yields correctly produced and placed parts.

The results of the client’s cycle are discussed in the following Section 6.1.5 to conclude the empirical research cycle.

6.1.5 Discussion of Results

This section discusses how the previously gathered knowledge questions and improvement goals (cf. Section 6.1.1) can be answered/addressed with the results of the client’s cycle.

Overall, the implementation at the client showed that errors in the process execution can be indeed detected in a real-world setting and help domain experts detect errors quickly.

Knowledge questions

The **first knowledge question** targets the ability to use process mining techniques easily in such settings. While the treatment at the client showed that process mining techniques can be applied easily on event streams and process execution logs in a real-world setting, the generation of a sufficient data set needed guidance by researchers. To accomplish the generation of an event stream using a process execution engine, events have to be reported back to the engine. Therefore notifications have to be sent to and from the engine to the

services, that are actually performing the tasks. At the client, this has been done using the HTTP protocol and web services. For tasks to be executed by human resources, an automatic way to log the execution of an event is preferable, like in Chapter 3.

After setting up a web service that responds to the process execution engine, domain experts by themselves produced the programs to generate the notifications needed to create an event stream. In addition, the domain experts independently created new process models and instances without the need of assistance.

Also for the **second knowledge question**, if domain experts can actually use the results of process mining techniques, the answer is split into parts. After a process model is created, the process execution engine ensures the correct enactment of tasks for a process instance for that model. Standard conformance checking focuses on the control flow of events, which is a time consuming task and provides results that should be ensured anyway. The client is satisfied with the advantages of creating an executable process model, but is more interested in the data flow of a process instance. Multi perspective conformance checking provides the client and the domain experts with knowledge on process instances going astray through temporal deviations, analyses the behavior and conformance of data elements inside and outside an event stream provided through external sensors. These results are highly appreciated at the client and help domain experts detecting problems in the execution of an automated process early on. At the moment, the results of conformance checking are only published as alerts. An implementation providing a visualization of deviations is desired as an addition to the framework.

Improvement goals

For the **improvement goals** defined in the research context, the implementation and execution at the client yielded promising results. An interesting observation has been made for the generation of suitable data sets. While the process execution engine is generating an event stream and a log for each process instance, the fragmentation of the process “Produce Turm Part” into many sub-processes, resulted into events that are assigned to their individual process instance, but the instance is not related to the production of a specific part. Therefore all spawned sub-processes by the main processes are now related to the main process to allow for better results using offline process mining techniques if desired. Another aspect that has been improved after the client’s cycle, is the separation of produced parts into individual process instances. As explained before, every spawned process is related to the main process in the event stream and log. However, for the interaction with the process execution engine, all production instances of one batch of parts, are designed as one process. This process spawns “Produce Turm Part” as often as parts are wanted, i.e., a loop is spawning the production process 5 times to produce 5 parts during one day. To distinguish between a spawned sub-process and the start of a new main process, a new icon is introduced for BPMN 2.0.

Fig. 6.6, shows the process for the production for one day. The process utilizes a custom BPMN event (intermediate throwing event, magnifying glass), which allows to group log-data based on individually produced parts, instead of saving it solely based on the given processes structure. While industrial processes are often modeled around the interaction of machines and the production of batches (restricted by raw material availability in the machine), the information derived from online mining is expected to be about individually produced parts. Thus when a process is modeled from a machine

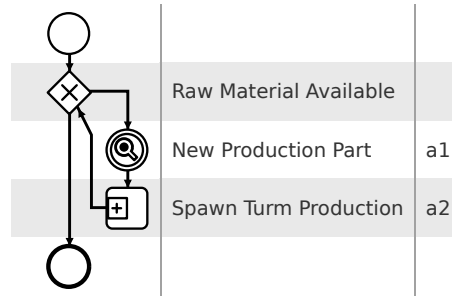


Figure 6.6: Process model for the daily production of parts. The signal indicates a new online mining item.

coordination point-of-view, a single instance contains information about multiple produced parts. The custom BPMN event is a simple way to tell where the event-stream for a single part, thus allowing to automatically extract and separate part-information.

The highlighting of deviating data elements is already explained in Section 6.1.4, where the alert of temporal deviations and the increased distance between two time sequences, helped detecting incorrectly executed process instances and discovering the source for the error. So both improvement goals (cf. Sect. 6.1.1) have been reached. The last improvement goal of highlighting deviating behavior in the execution order, cannot be fully answered at this moment. Since the process execution engine is ensuring the order of the tasks to be fulfilled, the main reason for deviating behavior is an error in the logging component or an attacker tampering with the process execution engine. None of these two scenarios has taken place in the study at the client. Since conformance checking for deviating behavior in the order of events is quite time consuming, this result can still show the benefit of having a process execution engine enacting process instances in an organization.

For the research goal, it can be concluded that an online process mining framework, indeed supports domain experts in an environment with defined process models. After the implementation, domain experts can model new process models, start and execute process instances. Instances containing errors in the execution are reported to the experts and offer an explanation for incorrect products. The implementation at the client yielded promising results. There are some threats to generalizability, since the manufacturing domain offers process models, which are usually already known by the organization and focuses on conformance checking. Other domains with increased human interactions which do not follow a strict structure could yield incorrect results.

Following the results of this TAR study, the impact of process mining in a company is presented in the next section through a focus group study [Kru14].

6.2 Expectations and Experiences of Process Mining in Action

To discover the impact of process mining in a company, the manufacturing domain has been chosen. [Rei20] presents best practice use cases and [CLS20] emphasizes the importance of process mining due to the data that is available in a manufacturing company. However, studies on process mining expectations and experiences in Small and Medium Sized

Manufacturing Companies (SMMC), are missing although SMMC account for 55.4% of manufacturing companies in the EU¹ and for 44.4% of the employees in manufacturing in the US². Moreover, these expectations and experiences have not been analyzed from the viewpoint of different organizational positions so far. It can be expected that due to the differences in daily work life as described below, expectations might vary which should be considered for a smooth introduction of process mining. In the following the different hierarchical positions are described in a SMMC for a better understanding of this study.

- ① *Shopfloor workers* tend to perform their work in a process-oriented way due to the structure of manufacturing processes, since a certain set of tasks has to be applied in a logical order. Even though most machines nowadays have their own logging mechanism, there is often no software orchestrating resources as well as coordinating the cooperation with other departments.
- ② *Supervising operatives* usually can observe specific steps in a process instance. If a workpiece or process subject is faulty due to an error, it is often unclear how and where in a process an error started occurring. Process mining can be vital for optimizing processes and detecting erroneous behavior.
- ③ For employees in *managing positions*, transparency is especially relevant. Transparency is a crucial aspect for companies nowadays, for legal protection as well as for cooperation with other companies. Process mining can increase the transparency by providing knowledge about business processes and their execution.

Using this scenario at SMMC and the algorithms from the previous chapters, the following questions haven been derived.

- What benefits and drawbacks are expected by SMMC when introducing process mining?
- What benefits and drawbacks are perceived by SMMC after the introduction of process mining?
- How can the implementation of process mining at SMMC be designed?

A focus group study following the guidelines stated in [Kru14] has been conducted to answer these questions. Focus groups have proven themselves as adequate means to assess the impact of process mining in practice [GMOvB20]. The specific study design for these questions is developed along a double layer approach enabling the distinction of the organizational position of participants and their exposure to process mining. The double layer approach is realized by two rounds of interviews with employees of two manufacturing companies covering organizational positions ①, ②, and ③. Moreover, in one company, process mining has already been introduced and the other is planning the introduction of process mining in the near future. Two real-world cases for process mining in manufacturing, i.e., electroplating and electronics assembly, are described in detail.

The findings of this study show that the expectations involve increased transparency which is crucial for collaborations with business partners. In addition, it is expected that

¹<https://ec.europa.eu/eurostat/statistics-explained/pdfscache/10086.pdf>

²<https://www.sba.gov/sites/default/files/advocacy/2018-Small-Business-Profiles-US.pdf>

process mining can help to detect deviations in process executions at runtime. Main concerns regard employees feeling observed by the increased transparency and reluctance of them to share tacit knowledge. The introduction of process mining confirms that the expected benefits indeed occur. Moreover, the decreased documentation effort for employees, due to process mining, outweighs the fear of surveillance of employees.

Section 6.2.1 explains the detailed structure of the focus group study and the participants. Section 6.2.2 introduces the real-world scenarios for process mining application in manufacturing. Section 6.2.3 contains a summarized overview of the results of the focus group interviews. The findings that can be deduced from the interviews are discussed in Section 6.2.4 where also future implications based on these findings are discussed and the research question answered.

6.2.1 Overview on Methodology and Study Design

This study employs focus groups [Kru14] to assess the expectations on and experiences with process mining in SMMC.

The focus groups are organized according to the double-layer design depicted in Fig. 6.7. The first layer distinguishes the focus group participants by their organizational positions, i.e., shopfloor worker, supervising operative, and manager. This distinction aims to identify the impact of process mining from different work perspectives. The second layer distinguishes the participants by exposure to process mining in their current company, i.e., if process mining has already been used in the company or not. Doing so aims at comparing the general expectations on process mining to its actual results.

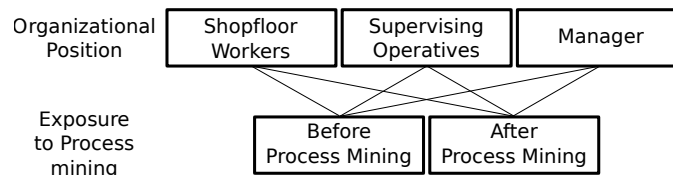


Figure 6.7: Double layer focus group study. All participants are grouped along both layers.

Two rounds of focus group interviews have been conducted. The first one consisted of three people who have not been using process mining in their work at the moment, but are planning to implement it in the near future. The second group consisted of four people, who are already using process mining, and plan to increase the usage of process mining.

As depicted in Fig. 6.7, participants of the focus group interviews can be distinguished along two layers.

The first layer focuses on the organizational position of a participant. In order to identify a set of participants for the focus group, a representative set of roles and their responsibilities based on [DLO00] is identified. As both companies operate in a lean teamworking environment but are SMEs and thus not necessarily differentiate roles as much as big companies, the set of roles down has to be reduced to a feasible number, that was then basis for organizing the actual focus group. ① reflects shopfloor workers who execute the tasks on the shopfloor. This task execution is then logged for applying process mining. Hence, the shopfloor workers can be seen as directly confronted with process mining and its results in their work life. ② reflects the supervising operatives of a company who are monitoring the shopfloor. Supervising operatives are interested in using

process mining to discover rarely executed paths in a business process, use conformance checking to detect faulty process instances and tasks that caused a failure. ③ reflects the manager of a department or company. Process mining can be used to evaluate the general performance.

The second layer of this focus group study emphasizes the exposure of process mining in the company. The participants are therefore split into two groups. ① of the second layer, reflects employees in a company which has not used process mining yet. The second group, ② consists of employees who are using process mining already. The associated process scenarios are introduced in Sec. 6.2.2.

6.2.2 Applied Process Mining Scenario

The study design outlined in Sec. 6.2.1 demands that selected participants of the focus group have already been exposed to process mining which is an important part of the interviews and the findings.

In the following, the scenarios, in which participants of the focus group (Shopfloor Workers & Supervising Operatives) have experienced the application of process mining, are introduced.

Electroplating

Company E produces parts which have to be surface-treated. This is done by submerging these parts in a chemical bath, giving them certain desired properties. After the bath is used for a certain amount of parts, or if the bath has been inactive for a certain time, it has to be refilled. For refilling, certain (dangerous) chemicals have to be combined. Before introducing a BPMN process-based orchestration solution to support the process, workers were following guidelines, taking notes, and manually filling out reports. In cases where these guidelines were not followed, accidents have occurred. Avoiding these kinds of accidents was one of the main reasons to introduce an orchestration solution.

After introducing a process-based solution, the process has been formalized as depicted in Fig. 6.8. The solution consists of two parts (CPEE [MRM14] BPMN notation): Fig. 6.8 (a) depicts a control process that determines based on sensors and human input, when to start a refilling cycle. Figure 6.8 (b) depicts the actual refill process, as carried out by two human workers. Figure 6.8 (b) starts with selecting a refill recipe. This can be either based on input from sensors in the bath or through human intervention from a supervising operative.

The recipe consists of a list of chemicals, and the required amount. Afterwards, the system waits for two workers to identify themselves through their NFC badges at the entrance of the chemicals storage locker. Only after their identity and role is established (one worker and one supervising operative are required), the locker can be opened. A screen shows which amount of which chemical has to be taken and added to the bath (in no particular order). Each chemical is in a container that is mounted on a digital weighting scale. Thus when the wrong amount of the chemical is taken, an emergency stop can be triggered. It is also possible to automatically track which chemicals have been used, as well as their exact amount. The workers are encouraged to write down their observations at a computer terminal after they are done (and the protective gear is removed).

In order to solve this problem, the assembly has been split into a number of sequential work packages, and for each work package a graphical worker assistance system has been designed. All logic for selecting individual steps and showing them on screen is implemented as a BPMN process-based solution. The worker assistance system automatically shows the correct set of steps for the work-piece in front of the worker (no variants have to be remembered), and also assumes a standard order of putting work-pieces together. Each step has to be acknowledged with a foot pedal. When a problem occurs, a worker can leave a (spoken, speech-to-text) note, and dismiss the work-piece for later fixing.

This setup forms a good basis for (online) process mining. It is possible to extract detailed information about durations and error rates, paired with information about the particular work-piece variation, used parts, and steps. Online process mining techniques are used to generate early warnings for supervising operators. Ex-post process mining is utilized for continuous process improvement. Though company E just started utilizing the system, early results have been deemed promising by workers, supervising operators, and management.

6.2.3 Results of Focus Group Interviews

The double layer design of the focus group study is depicted in Fig. 6.7 and explained in Section 6.2.1. The focus group features two interviews with employees from manufacturing companies CDP and E.

Manufacturing company CDP: The first focus group contained participants of two management levels. Three participants were interviewed, i.e., one supervising operative, and two general managers/CEOs. None of them was using process mining in their department at the time of the interview.

Manufacturing company E: The second focus group contained participants of three management levels. Company E is in the metal-processing domain and employs around 750 people. Four participants have been interviewed, i.e., two supervising operatives, a general manager and one shopfloor worker.

After an introduction into process mining, all participants revealed a good understanding of the basic principles of process mining and could identify scenarios in their company, where process models are already in place, i.e., in the electroplating department (cf. Sec. 6.2.2).

Table 6.1 provides a summary of the profile of the participants to identify their answers. In the following, the interview results are presented for each question.

What benefits do you see in a process-oriented view of your field?

ED thinks that one's workload is better structured using a process-oriented view, which increases the cooperation quality with other departments. The operative CA, sees benefits of process mining with respect to the transparency of their department and their company. Knowledge, in particular, domain specific knowledge is lost if an employee leaves, is a concern mentioned by CA. Workflows here are not explicitly available as formal models, but workers loosely follow learned rules/guidelines, hence it is difficult to detect the source of an error. Conformance checking and process model discovery are regarded as useful techniques to ease these problems. These benefits are confirmed by the operatives EA and EB. The correct execution of a process instance, supervised by process mining, allows them to detect and react to errors as soon as they happen. Moreover, the process models enable a good visual representation of the currently active tasks. EB mentions that, "A huge

Table 6.1: Focus Group Participants Profile

Coding Participants	Company	Position	Experience in Company	Working with Process Mining
EA	E	Supervising Operative	>10 years	yes
EB	E	Supervising Operative	2-4 years	yes
EC	E	General Manager	>10 years	yes
ED	E	Shopfloor Worker	>10 years	yes
CA	CDP	Supervising Operative	>3 years	no
CB	CDP	General Manager	>2 years	no
CC	CDP	General Manager	>4 years	no

advantage of a process-oriented view is the improved communication between employees from all levels”.

The managers, CB and CC, share concerns regarding the usability of process mining in the daily routine of employees. The discovery of process models is seen as an important feature of process mining as it increases transparency, which is often required for cooperations with other companies. The correct process execution is crucial as well, to discover and fix problems. EC confirms the previously outlined benefits. In addition, the application of process enhancement is envisioned in the near future, through implementing lean management techniques and optimizing resource sharing between multiple departments.

How are processes and tasks executed and logged at the moment?

ED states, that their department uses work instructions as a basis for processes, obtained by interviewing workers. It was mentioned that this is useful for new employees, but yields some uncertainties (e.g., for rarely produced parts). CA explains that most of the activities are still logged manually in a rudimentary way without much information on the input / output of each task. The detection of faulty behavior in the process execution is crucial, but hard to track without a rich documentation. EA mentions, that unlike the electroplating unit, in his unit everything is currently only logged in an ERP-system. However, these event are only available at a high level and only for certain tasks, e.g., only measurements are logged, but not the production itself. These logs are used for making operational decisions, such as determining the delivery date. EA is aware that this leads to resource waste, as parallel processes are not properly synchronized, and departments sometimes have to wait on other departments, because they decided on a sub-optimal production order. EA also claims, that the work instructions mentioned by shopfloor workers, are often not followed, but instead slight variations learned from colleagues are used. CB emphasizes again, that identifying errors and increasing the efficiency is very important. Therefore, processes have been modeled showing the interaction between humans. These interactions are currently logged in an ERP system. Process mining techniques such as conformance checking or using a system to enact the correct tasks at the right time have not been used. EC is aware of the benefits of process mining in the implemented scenarios. Additionally EC mentioned the wish to implement process mining at the managerial level, i.e., mine and analyze management processes.

How is the correct execution of a process model currently ensured?

Process models are used and tasks are logged with a process execution engine in the application scenario of the electroplating unit as mentioned by ED. Currently active tasks are shown on a screen and are executed by interacting with the screen. CA, CB, and CC state that, as no process models are used, their correct execution is not ensured. EA explains that, correctness for the scenarios is enforced by a process engine, but for many other scenarios, the status quo has not changed. EB says, that additional process mining techniques to automatically notice errors is desirable, as currently root-cause analysis for errors is mostly done manually. EC is aware of the benefits of process mining in the implemented scenarios as decisions regarding high-level process changes become easier, and controlling is improved. EC again states that processes at managerial level should be formalized as well.

Which advantages do you see for your company with the support of process mining?

ED sees a reduced documentation effort due to automatic documentation. The instructions are well presented and help following the process model. CA emphasizes the importance of process enhancement as an important factor in the company, but is also keen on improving the efficiency using process mining techniques in general. EA sees a lot of potential, especially for protection against insurance claims if accidents happen or if products do not adhere to the quality standards. EB mentions that with increased process standardization they would be able to take on more risky projects. EA mentions an accident that happened in a sub-department where the cause could not be determined. To avoid such accidents in the future, it is essential to better structure the workflow, making it more transparent, provide support for the employees taking part in critical processes and log interactions with dangerous chemicals. CC sees advantages in understanding of processes for different positions in the organizational hierarchy. CB also thinks that processes can be communicated better between companies from different domains for a more efficient cooperation. With the help of process mining, especially process discovery and conformance checking, the perspectives of the shopfloor level and the management level should be more aligned. In the company, workflows rarely show deviations and more often follow a common path, which should allow for understandable process models. CC mentions explicitly that *“While a performance evaluation of a process can be done every three months and does not have to be online, a deviation of a process instances should be reported immediately”*. EC added, that there are additional benefits for planning and analysis that could be obtained by introducing process mining.

Which advantages do you see for your specific department with the support of process mining?

ED sees a big advantage, in the training of new employees with the use of process models and process mining. Process models provide a good visual representation of the workflow and allow for a better communication between departments. Online process mining can give immediate feedback about the current state of produced parts. CA points out the importance of identifying errors and the increased efficiency when communicating with other departments based on data produced by process mining. Both operatives, EA and EB, think that process transparency is increased due to the use of process models and a

process execution engine. They mention automatic reports after each crucial step executed by shopfloor workers, which help to ensure the conformance of a process instance (regarding many aspects: process structure, timing, resource deviations, data deviations). **CA** emphasizes that not only the production should benefit from process mining techniques, but tasks involving only humans as well, such as creating reports, delivering a product, and communication between departments. **EC** again emphasize that data obtained through process mining (e.g., duration & resource utilization for a multitude of product variations) are a huge benefit for planning and process optimization.

What problems do you anticipate for the introduction of process mining in your department?

ED sees the benefits of process mining in one's department, but fears that long-term employees still might not see the purpose of process mining in other departments, because they are often not interested in changing their daily routine. However, **ED** states that if the benefits, i.e., less documentation effort, are clear to the employees, they can be convinced. **CA** voices concerns about the acceptance by the workers, since they tend to use their acquired knowledge to secure their position in the company. **CA** also fears high costs for heterogeneous workflows, since the discovery of the process model and its variants could imply a huge effort. The advantages of process mining are clear in **CA**'s opinion. **EA** fears that the employees could feel observed. Hence, **EA** thinks it is important to encourage strong involvement of employees when implementing future scenarios. **CC** echoes the concerns about employee acceptance. The increased process transparency is viewed as critical, as it paves the road towards cooperations with future customers and partners. **CB** voiced concerns, that the increased logging and data availability makes data leaks possible, which would harm the company. **EC** thinks that employee acceptance is a challenge, but in hindsight was easier to achieve than expected. **EC** thinks that the introduction for the whole company is too complex and that they will aim for implementing process mining in many small projects (as they want to focus on techniques that require heavy use of domain knowledge –analysis of process data, durations and resource usage). Lastly, **EC** raises the concern that the current IT infrastructure (networking and computational power– more sensors produce more data requires more analysis capabilities) and human resources are not sufficient. Currently, process mining has been successfully introduced in one department.

6.2.4 Discussion and Implications for Research and Practice

Based on the results of the focus group interviews as summarized in Section 6.2.3, the following findings have been deduced. The findings can be categorized as follows:

- Requirements before process mining can be introduced.
- Expected results when introducing process mining.
- Actual improvements after process mining has been introduced.

The three categories are now discussed in detail.

Requirements

The settings in both companies CDP and E distinguish themselves by the granularity of the logged tasks. The first focus group from CDP does not use any of the three fields of process mining at the moment, but is already working with the support of a process execution engine, which enables the creation of an event stream and the automatic documentation of each task in a process. In company E, by contrast, not every task is logged, but only certain checkpoints, i.e., a finished piece. This leads to inaccurate process execution logs, since it is not clear, how and when the different tasks have been executed. Company E is using a process execution engine only in a sub-department. In other departments of company E different approaches have been tried, i.e., a manually created handbook of business processes for new employees. Unfortunately, this handbook is rarely used and instead knowledge is transferred from senior employees to newer ones. This leads to undocumented steps, which renders retrieving fine granular results and therefore process mining on a more granular basis impossible.

The focus group interviews showed, that even though companies are putting effort in creating process models through intensive interviews with employees and are making these process models available, the documentation of tasks is often too time consuming. However, the introduction of process mining supported by a process execution engine showed, that employees are willing to log their tasks if enough support is available, like a monitor showing the current active task and an automated documentation. The supervising operatives and managers are benefiting from the generated reports about conformance of a process instance and general behavior through process mining.

Expected Results

Most of the participants share similar experiences concerning the process of creating process models i.e., through interviews, since employees often follow a process from tacit knowledge. Since it is important to be as transparent as possible for potential business partners as per the statements of the focus group participants, a better representation of the actual processes is desired. Another important factor concerns correct process execution as this increases transparency. The participants also emphasize the moment of time when a deviating process instance is detected. While the evaluation of a whole department can be calculated every few months, a process instance with a deviating conformance should be detected as early as possible. To check the conformance during execution, an event stream is required to apply process mining.

For the implementation, the participants raised concerns about the introduction of process mining in their departments. Employees could feel observed, since their daily routine could be analyzed from the process execution logs. Another problem is, that employees sometimes tend to gather knowledge and not share it, making themselves harder to be replaced. The participants agreed, that the employees should be involved in the process of introducing process mining. It was also mentioned that as soon as the benefits of the approach became very clear, acceptance was very high.

In addition, it was mentioned, that the IT infrastructure could be an issue for implementing process mining.

Improvements

The introduction of process mining in a department of company E results in the following improvements. The process of obliging two employees to perform several tasks, where one of them has to have a specific role, can be accurately logged with the support of a process execution engine. Conformance checking, taking the data perspective into account, can reveal deviations, if the criteria of the correct amount or the correct roles is not fulfilled. Another important aspect is the temporal perspective. Conformance checking allows to detect temporal deviations in the process, e.g., an extremely short duration for putting the protective gear on, leading to the assumption that the gear is not worn correctly.

When a deviation is detected at runtime, it is possible to provide the company with the information for which process instance the deviation occurred. With this information, it can be tried to explain the reason for this deviation through the information stored for a process instance by the process mining framework.

Based on the findings, creating automatic reports to detect undesired behavior in process instances and help to ensure the correct order of events is beneficial. Drawbacks such as the fear of surveillance can be avoided through outlining major benefits of process mining to shopfloor workers, including the automatic documentation of tasks.

As to how the implementation of process mining in SMMC can be designed, as mentioned in Section 6.2.3, a process model is often already available in the production, generated from the knowledge of the shopfloor workers and process supervisors. Based on the interviews, it can be concluded that correct process execution and its documentation are of utmost importance. This can be achieved by implementing and executing the existing process model through a process execution engine. The engine is used to orchestrate active process instances of process models and manages the documentation of tasks, i.e., timestamps of start and end events. To give shopfloor workers a better visualization of the process and the currently active task in a process instance, a screen can be used to provide additional information. Utilities, such as a hand scanner or a foot pedal, can be used to automatically complete the current task in a process instance which leads to the next task shown to the worker. A possible setup is the Electroplating process (cf. Sec. 6.2.2). To increase the knowledge of currently active process instances, wearable information systems can be connected to the process mining framework as well and display process instances not matching the expected behavior [SJEA18].

Discussion

When looking at the significance of the results, three groups can be established.

Not surprising: Digitalization gaps exist and SMMC struggle to close them. All participants agree that explicit process orchestration from the business level to the shop-floor level will improve the quality of available event logs, and is a first step towards online process mining and process enhancement. It became clear that SMMC suffer from a lack of IT resources. However, they are aware that process mining and data analysis in general will help them with digitalization (i.e., new ways of interacting with their customers).

Expected, but disappointing: Process discovery is not considered important. All participants agreed that process elicitation through explicit modeling leads to better results and understandability. This was not unexpected as SMMC often have flat hierarchies, hence involvement and knowledge of the processes is high. The participating companies

(some of the participants also talked about previous employments) often utilize flexible manufacturing islands with unstructured manual labor instead of production lines. The effort for data collection there could very well be so high that focus group participants might be right.

Surprising: Shop-floor workers were expected to be critical of process mining supervising operatives and management alike. However, they were very easily convinced when demonstrating process mining results. Supervising operatives and management wish for the application of process mining on high-level processes, but can neither clearly express the expected results nor have a clear vision how to digitalize these processes. Conformance checking is well understood by the focus group participants. Mining of temporal deviations and performance indicators based on fine-grained sensor data are seen as an important short-term goal. Surprisingly, online process mining, i.e., making deviations visible and explainable at runtime, is considered more important than ex-post analysis.

Limitations and Threats to Validity

Focus group interviews bear certain threats to validity [Kru14]. In particular, investigating expectations and experiences of process mining in SMMC is relatively complex. Hence, there is a threat of either made up answers, i.e., caused by insufficient experience of a participant or trying to avoid negative feedback by colleagues afterwards, or just trivial answers caused by too many participants. To minimize these threats, small focus groups have been chosen, ensured that a certain level of knowledge of processes in general is present, and the questioning route following the guidelines in [Kru14] has been developed. Further limitations involve:

- *Transferability to other domains:* Manufacturing can be seen as “killer application”. Hence it is promising to look at other domains such as medicine that also combine processes, physical world, and human work.
- *Generalizability:* SMMC struggle with specific problems, hence the generalizability to bigger companies is questionable. Moreover, while a small focus group helps in getting meaningful results for complicated subjects, it can still be argued that similar SMMC are not sharing the same experiences. More interviews with different SMMC could overcome this limitations.

Finally, the companies and participants of the focus group were all volunteers, that answered to an email to a list of companies that regularly participate in research projects. It is possible that (a) the results are not representative of SMMC, or (b) a John Henry effect (over-performance) [Sar72] regarding process mining was observed.

6.3 Conclusion and Outlook

This chapter focuses on the execution of a process and analyzes the behavior of process instances to a process model.

A possible solution design is presented in Section 6.1, the experimental framework TIDATE following a Technical Action Research study.

The implementation at the client

- revealed differences section a real-world setting and the academic environment, in particular, the different perspectives on process instances, i.e., batch-oriented vs.

product-oriented.

- led to answers to research and improvement questions, for example, how process mining can be integrated in the manufacturing domain and if domain experts can use these techniques and with how much effort required.

The introduction of a process execution engine and the nature of the manufacturing domain, helped applying process mining and the results look promising. Time deviations and deviations in data elements, even from external sources, like data sensors, are providing much needed information on the conformance of a process instance and give a close inspection to analyze where a process instance is deviating from the expected behavior.

Since not every domain has the same degree of process orientation as the manufacturing domain, conducting a TAR study with TIDATE in another domain is desired.

The focus group study in Section 6.2 collected expectations on and experiences with process mining in SMMC, including two real-world process mining scenarios at one company's side. The main findings are that a suitable data set generation is a major challenge (I), transparency of business process becomes increasingly important (II), Human resources should be included into the process (III), and that Infrastructure plays an important role for SMMC (IV).

The status quo in SMMC is that logging is part of the business logic and data-centric (I). Selected milestones in the production produce a data dump with a timestamp, while most process steps in the manufacturing domain just produce no events at all.

Transparency (II) is considered important for four key aspects: (a) legal protection against insurance claims, (b) protection against liability claims when dealing with bad parts, (c) reduction of erroneous parts before quality control, and (d) streamlining of processes when dealing with a huge number of product variants in combination with human resources.

It is important to consider that human resources may feel observed and become reluctant to share their tacit knowledge (III). Successful communication and demonstration of the benefits of process mining, on the other hand led to high acceptance among workers.

Currently, the local IT infrastructure is a perceived bottleneck (IV) for the increasing data volume and velocity that comes with fine-grained logging of all steps involved manufacturing and production of goods.

Another important aspect for future work is how the information on deviations is conveyed to the clients, i.e., domain experts. In this study, simple alert notifications are sent if a deviation is detected, but a visualization of the deviating tasks of the process model in the process execution engine, would lower the entry barrier even more for domain experts to get accustomed to process mining.

7 Conclusion

This chapter provides a summary of this thesis. Section 7.1 provides an overview of the results of the Chapters 3 to 6 where each research question of the thesis is addressed. Potential future work is described in Section 7.2.

7.1 Results of Research Questions

The results of the different chapters are here summarized and used to answer the research questions of this thesis, defined in Section 1.3. A short summary of the chapters is described in the following:

- Chapter 3 established a new serialization format for XES data, an approach to create an event stream and process execution files online, as well, as a way to integrate human resources into the automatic generation of such files.
- Chapter 4 presents a novel approach for detecting drifts in the workflow perspective as well as the data perspective at runtime and defines process histories, a collection of the process models between each drift.
- Chapter 5 focuses on conformance checking algorithms and establishes an advanced cost function for adjusting the costs of structural deviations, a new approach to handle semantic deviations in the temporal perspective, as well as explaining the root cause of concept drifts based on external sensor streams, e.g., external deviations.
- Chapter 6 gives an overview on the acceptance of the presented algorithms as well as on the expectations companies have related to process mining.

7.1.1 Creating Data Sets for Process Mining

Chapter 3 focused on the following research questions

- RQ ①a How should a data format for process mining algorithms be designed to be created during the execution?
- RQ ①b How to generate data sets directly at the execution of a process for online process mining algorithms?
- RQ ①c How to improve the quality of the data set and reduce the error-proneness?

Section 3.1 addresses RQ ①a. For a suitable data format, different operations have been identified, which are crucial for the performance of algorithms using the data format. A human readable format is advantageous, since it can be easily modified by all ubiquitously available tools and also is easy to handle for domain experts. Consequently, the data

set format should allow to comment on log entries for a better understanding. For the automatic handling of a data set for process mining algorithms, a defined structure, which can easily be checked by using a specific schema is mandatory.

The most crucial aspect in an online setting is the performance, i.e., how long it takes to generate a new log entry. Different file formats have been analyzed, leading to the answer that, while XML provides a solid serialization format for the XES format, it is not the format for creating such XML-XES files during the execution, since the format is tree based, and for a new log entry the complete tree has to be parsed, saved in memory and generated again. YAML proved to be a valid choice for the XES serialization, since it is able to save the same information as XML and it is possible to append new log entries easily to already existing log files, without parsing the file.

Section 3.2 addresses RQ (1b). For online process mining algorithms, an event stream is used as an input source. These events should be generated when an activity in a process instance is executed. This led to the use of a process execution engine, which orchestrates the currently active activities and generates and injects events into a stream. For the specific lifecycle transitions of an activity, specific events in the process execution engine have been defined, which allows process mining algorithms to easily filter the event stream. The presented approach creates events for resources without human interaction automatically during their execution. Events with human interaction are not yet taken into account, which is achieved using the approach to answer RQ (1c).

Section 3.3 addresses RQ (1c). To improve the data quality, especially with human resources, a process execution engine is used in conjunction with a new NFC approach. Each time an activity is executed, the human resource can apply an NFC tag to an NFC reading device. This reading device is connected to the process execution engine and automatically generates a new log entry for the data set. The advantage of this approach, is that the timestamp of an event is accurate and events are not lost, because in current scenarios, human resources tended to generate the log entries of their activities at the end of their shift, thus forgetting sometimes some events. The NFC approach reduces the amount of time spent on generating log entries and generate data sets of a higher quality.

While addressing RQ (1a)-(1c), the following lessons were learned:

Preparation of Event Stream. For online process mining algorithms an event stream is required, but generating such a stream at the point of time an activity is executed proved to be challenging. Even with the help of a process execution engine, data elements, like acting resource, have to be present during all the different lifecycle events to be cohesive, which depending on the engine is not always easy to achieve. Another challenge proved to be the timestamp of an event. Even if a process execution engine is creating an event directly at the point of time of the execution, it is still possible to arrive in wrong order at the end service. This is often related to connection issues, e.g., two events are created nearly at the same time, but due to another process finishing at one service, the other event can slightly arrive earlier at the end service. A timestamp created at the end of the as a data element in the event can be used instead, to mitigate this behavior.

Acceptance of Human Resource. Through interviews with domain experts in the care domain, the importance of involving human resources became evident. Staff members in the care domain have to document every activity, which results in a lot of documentation work. Thus a solution for creating data sets for process mining algorithms should not increase the workload of human resources. The NFC solution presented in Section 3.3 enables human resources to generate events directly when an activity is executed without

increasing the workload. An additional benefit of this approach is an automatically generated documentation for staff members in the care domain, which motivates users to interact with the system.

7.1.2 Discovering the Evolution of Processes through Concept Drifts

Chapter 4 focused on the following research questions

- RQ ②a How can the evolution of a business process be discovered at runtime?
- RQ ②b How do data elements relate to concept drifts?

Section 4.1 addresses RQ ②a. Processes evolve over time, i.e., the logic of a process changes due to a new legislature for example. These process changes are not always communicated to every layer in a company, e.g., the execution of a process changes, but the process model to reflect the changes is not updated. The solution in Section 4.1 provides a definition of process histories, which effectively uses an event stream to discover a new process model as soon as the events of an event stream are not fitting the currently used process model. The collection of the discovered process models defines a process history. The change between two different process models is described and defined in four different types of concept drifts based on a process history, i.e., a change in the workflow perspective of a process, similar to [BvdAŽP11].

Section 4.2 addresses RQ ②b and focuses on changes in the data perspective between two different process models in a process history. To achieve this, the data elements attached to events in an event stream are collected. For each discovered process model in a process history, statistical values, e.g., the mean and standard deviation for numerical data elements or the occurrences of different roles for resources, are collected and added to the process model. The elaborated algorithms analyze the data elements in the event stream for outliers. If the data elements are not fitting the values from the process model, a data drift occurred and a new process model is discovered with updated values for the data elements in the model. Data drifts can be distinguished by four different type of drifts, similar to concept drifts.

While addressing RQ ②a-②b, the following lessons were learned:

Types of Concept Drifts. While four different types of concept drifts are already known, a formal definition has been missing. When defining the four types of concept drifts based on a process history, it became evident, that one type is not automatically excluding another type, e.g., a recurring drift, can also be a gradual drift for example, if there are still some process instances active following the behavior of the previous model. The recurring drift can also be a sudden drift instead, if process instances following the behavior of the previous model are aborted. The definition of the concept drifts hence enables to discover different combinations of concept drift.

Different Drifts can happen simultaneously. Concept drifts usually are related to the workflow perspective of a process model, i.e., the order, the addition, and the removal of events in a process. An interesting finding is that, drifts, that occur in the data perspective, can simultaneously occur with a drift in the workflow perspective. A new legislation, for example, may add a new event to a production process and, in addition, enforces events of this process to be executed with by a resource with another role, e.g., a

doctor instead of a nurse. This reflects an incremental drift in the workflow perspective and a gradual drift in the data perspective.

7.1.3 Time & Data-Aware Conformance Checking and Explaining Drifts

Chapter 5 focused on the following research questions

- RQ ③a How to better quantify the costs of deviations between process instances and a process model using data elements?
- RQ ③b How can the temporal perspective be taken into account for conformance checking?
- RQ ③c How to discover the source of a concept drift?

Section 5.1 addresses RQ ③a. Deviations in the behavior of the process execution and the behavior of a process model can be detected at different levels. This section concerned itself with deviation at the structural level, i.e., missing or additional events in a process instance. Conformance checking algorithms assign a cost to a deviation, but the standard cost function is not differentiating between events. This section introduces the advanced cost function for conformance checking algorithms which adjusts the cost of a deviation based on the data elements of an event. If the data elements contain correct values, even though some events are missing, the costs for a deviation can be reduced.

Section 5.2 addresses RQ ③b. In this section, semantic deviations are analyzed, in particular temporal deviations. There are two important temporal distances to observe. First, the distance between two different events and the distance between the start and the end of an event, i.e., the duration of an event. A new conformance checking algorithm is presented to detect temporal deviations, for which a cost is assigned based on the severity of the deviation.

Section 5.3 addresses RQ ③c and focuses on external deviations. Process instances can be influenced by data streams, that are not available in a process instance, e.g., the temperature and humidity of a room or the current power consumption of a resource. These data streams are generating data points constantly, but are not typically assigned to a specific event in a process. The provided algorithms relate process instances to specific intervals in the data streams and explain the reason for a concept drift in a process by identifying drifts in the external data streams through dynamic time warping algorithms.

While addressing RQ ③a-③c, the following lessons were learned:

Areas of Deviations. Following the results of each section in this chapter, different types and areas of deviations in conformance checking algorithms have been identified. Traditional conformance checking algorithms, focus on structural deviations, i.e., the addition and absence of events in a process instance. Semantic deviations are discovered by taken the data elements of events into account [MdLRvdA16a]. Section 5.2 has taken a new approach for detecting semantic deviations by focusing on temporal deviations, i.e., the time between two events or the duration of an activity. The external data, e.g., data streams that are not related to a process instance directly, is addressed by identifying deviations in data sensor streams using time series processing algorithms, i.e., dynamic time warping.

External Data Streams relating to Process Instances. The external data streams are usually constantly producing data points in the streams and are not assigned to a specific event, i.e., the blood pressure of a patient is measured constantly without a fetching event in the process instance, which reads the blood pressure measurements each millisecond. Thus it is important to discover the snippets of a data stream that are relevant for specific events in a process instance to apply conformance checking algorithms onto and to discover the root cause of a concept drift.

7.1.4 Evaluating TIDATE - Time and Data Aware Process Mining at Runtime

Chapter 6 focused on the following research questions

- RQ ④ What are the general expectations on process mining algorithms by domain experts and what are the actual results after it has been introduced?

Sections 6.1 and 6.2 address RQ ④. To answer this question, the presented algorithms in this thesis are evaluated as an artifact and evaluated using the technical action research methodology in Section 6.1. A focus group study is conducted in Section 6.2. The participants are categorized in two layers, differentiated by their hierarchical position in their company and their level of exposure to process mining. It was confirmed, that process mining is a valuable asset for domain experts and that they request results as soon as possible, i.e., online process mining instead of offline process mining.

While addressing RQ ④, the following lessons were learned:

Transparency. The focus group interviews showed, that transparency is of utmost importance for companies. It is important for their own optimizations and it is required for a cooperation with other companies nowadays. Thus process mining proofs to be an important asset.

Inclusion of Employees. Employees tend to be afraid of changes in a company and are afraid of sharing knowledge of the companies work flows with other colleagues to make them not replaceable. Thus it is important to include employees and a possible works committee in the project of introducing process mining algorithms into a company.

7.2 Future Work

This thesis provides algorithms to generate a data set for process mining algorithms in general, to collect the history of a process, and to compare the behavior of a process model to the behavior of a process instance. Conformance checking algorithms are now able to adjust deviations based on an advance cost function and can take data elements into account. Deviations in external data streams open a new field used for explaining the root cause of concept drifts. Thus, a process history provides new insights into the evolution of a process and could possibly be used to predict future changes. Future work is identified in the two following categories:

Exception Handling: The algorithms in this thesis are able to detect deviations in a process instance. A deviation can reflect an error in the process execution which can be handled by different exception handling strategies, e.g., if a resource is faulty, new process instances could use a different resource. Different exception handling strategies need to be

identified, described, and algorithms need to developed to handle these strategies. The strategies are then to be incorporated into a process history.

Temporal Profiles: Temporal profiles are described in Section 5.2, but currently, temporal profiles are generalized for all process instances and neglect data elements. The temporal profile for patients in a hospital could vary greatly for patients, that are 20 years old, to patients that are 70 years old. Thus, it seems feasible to discover more specialized temporal profiles depending on specific data elements. Future work is aimed at analyzing the impact of specialized temporal profiles.

Moreover, future work is also aimed at developing better algorithms for incorporating multiple different data sensor streams to discover the data sensor streams, which are the root cause of a drift and use these explanations to predict future process changes based on the changes found in a process history.

Bibliography

- [AB12] Pankaj Agrawal and Sharad Bhuraria. Near field communication. *SET-Labs Bridfings*, 10(1):67–74, 2012.
- [Adr14] A. Adriansyah. *Aligning observed and modeled behavior*. PhD thesis, Department of Mathematics and Computer Science, 2014.
- [Agg07] Charu C. Aggarwal, editor. *Data Streams - Models and Algorithms*, volume 31 of *Advances in Database Systems*. Springer, 2007.
- [Alp20] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.
- [AvZQ⁺21] Jan Niklas Adams, Sebastiaan J. van Zelst, Lara Quack, Kathrin Hausmann, Wil M. P. van der Aalst, and Thomas Rose. A framework for explainable concept drift detection in process mining. In Artem Polyvyanyy, Moe Thandar Wynn, Amy Van Looy, and Manfred Reichert, editors, *Business Process Management*, pages 400–416, Cham, 2021. Springer International Publishing.
- [AZAMBH18] Laila Akhu-Zaheya, Rowaida Al-Maaithah, and Salam Bany Hani. Quality of nursing documentation: Paper-based health records versus electronic-based health records. *Journal of clinical nursing*, 27(3-4):e578–e589, 2018.
- [BC94] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.
- [BC17] Andrea Burattin and Josep Carmona. A framework for online conformance checking. In *International Conference on Business Process Management*, pages 165–177. Springer, 2017.
- [BCD15] Anne Baumgrass, Cristina Cabanillas, and Claudio Di Ciccio. A conceptual architecture for an event-based information aggregation engine in smart logitics. In *Enterprise Modelling and Information Systems Architectures*, pages 109–123. Gesellschaft für Informatik eV, 2015.
- [BFN⁺19] Michael Borkowski, Walid Fdhila, Matteo Nardelli, Stefanie Rinderle-Ma, and Stefan Schulte. Event-based failure prediction in distributed business processes. *Inf. Syst.*, 81:220–235, 2019.
- [BGvdW09] Melike Bozkaya, Joost Gabriels, and Jan Martijn van der Werf. Process diagnostics: a method based on process mining. In *2009 International Conference on Information, Process, and Knowledge Management*, pages 22–27. IEEE, 2009.

- [BKEI05] Oren Ben-Kiki, Clark Evans, and Brian Ingerson. Yaml ain’t markup language (yaml™) version 1.1. *yaml.org, Tech. Rep*, page 23, 2005.
- [BM98] BA Brooks and K. Massanari. Implementation of nanda nursing diagnoses online. north american nursing diagnosis association. *Computers in nursing*, 16(6):230–6, 1998.
- [BRK20] Lucas Baier, Josua Reimold, and Niklas Kühl. Handling concept drift for predictions in business process mining. In *2020 IEEE 22nd Conference on Business Informatics (CBI)*, volume 1, pages 76–83. IEEE, 2020.
- [BSvdA14] Andrea Burattin, Alessandro Sperduti, and Wil MP van der Aalst. Control-flow discovery from event streams. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2420–2427. IEEE, 2014.
- [BvdAŽP11] RP Jagadeesh Chandra Bose, Wil MP van der Aalst, Indrė Žliobaitė, and Mykola Pechenizkiy. Handling concept drift in process mining. In *International Conference on Advanced Information Systems Engineering*, pages 391–405. Springer, 2011.
- [BVDAZP14] RP Jagadeesh Chandra Bose, Wil MP Van Der Aalst, Indre Zliobaite, and Mykola Pechenizkiy. Dealing with concept drifts in process mining. *IEEE transactions on neural networks and learning systems*, 25(1):154–171, 2014.
- [BvZvdA19] Alessandro Berti, Sebastiaan J van Zelst, and Wil van der Aalst. Process mining for python (pm4py): bridging the gap between process-and data science. *arXiv preprint arXiv:1905.06169*, 2019.
- [CA86] Linda Crocker and James Algina. *Introduction to classical and modern test theory*. ERIC, 1986.
- [CCL07] Harry K.H. Chow, K.L. Choy, and W.B. Lee. A dynamic logistics process knowledge-based system - an {RFID} multi-agent approach. *Knowledge-Based Systems*, 20(4):357 – 372, 2007.
- [CG01] Scott Saobing Chen and Ramesh A Gopinath. Gaussianization. In *Advances in neural information processing systems*, pages 423–429, 2001.
- [CGG⁺07] Robert C Cannon, Marc-Oliver Gewaltig, Padraig Gleeson, Upinder S Bhalla, Hugo Cornelis, Michael L Hines, Fredrick W Howell, Eilif Muller, Joel R Stiles, Stefan Wils, et al. Interoperability of neuroscience modeling software: current status and future directions. *Neuroinformatics*, 5(2):127–138, 2007.
- [CLRS09] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.
- [CLS20] Angelo Corallo, Mariangela Lazoi, and Fabrizio Striani. Process mining and industrial applications: A systematic literature review. *Knowledge and Process Management*, 27(3):225–233, 2020.

- [Cro06] Douglas Crockford. The application/json media type for javascript object notation (json). *RFC 4627*, 2006.
- [CvDSW18] Josep Carmona, Boudewijn F. van Dongen, Andreas Solti, and Matthias Weidlich. *Conformance Checking - Relating Processes and Models*. Springer, 2018.
- [CW99] Jonathan E Cook and Alexander L Wolf. Software process validation: quantitatively measuring the correspondence of a process to a model. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 8(2):147–176, 1999.
- [DHK⁺10] Diane M Doran, R Brian Haynes, André Kushniruk, Sharon Straus, Jeremy Grimshaw, Linda McGillis Hall, Adam Dubrowski, Tammie Di Pietro, Kristine Newman, Joan Almost, et al. Supporting evidence-based practice for nurses through information technologies. *Worldviews on Evidence-Based Nursing*, 7(1):4–15, 2010.
- [DLO00] Rick Delbridge, James Lowe, and Nick Oliver. Shopfloor responsibilities under lean teamworking. *Human relations*, 53(11):1459–1479, 2000.
- [dMvdAR15] Eduardo González López de Murillas, Wil MP van der Aalst, and Hajo A Reijers. Process mining on databases: Unearthing historical data from redo logs. In *International Conference on Business Process Management*, pages 367–385. Springer, 2015.
- [DMvDVdAW04] AK Alves De Medeiros, Boudewijn F van Dongen, Wil MP Van der Aalst, and AJMM Weijters. Process mining: Extending the α -algorithm to mine short loops. 2004.
- [DRMGF14] Reinhold Dunkl, Stefanie Rinderle-Ma, Wilfried Grossmann, and Karl Anton Fröschl. A method for analyzing time series data in process mining: application and extension of decision point analysis. In *CAiSE Forum*, pages 68–84, 2014.
- [DRMR18] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management, Second Edition*. Springer, 2018.
- [Dro17] Michelle Drolet. How much will non-compliance with gdpr cost you? CSO, October 2017.
- [DvC⁺16] Claudio Di Ciccio, Han van der Aa, Cristina Cabanillas, Jan Mendling, and Johannes Prescher. Detecting flight trajectory anomalies and predicting diversions in freight transportation. *Decision Support Systems*, 88:1–17, 2016.
- [EFMR19] Matthias Ehrendorfer, Juergen-Albrecht Fassmann, Juergen Mangler, and Stefanie Rinderle-Ma. Conformance checking and classification of manufacturing log data. In *Business Informatics*, pages 569–577, 2019.

- [FHBV11] J. Fontecha, R. Hervas, J. Bravo, and V. Villarreal. An nfc approach for nursing care training. In *2011 Third International Workshop on Near Field Communication*, pages 38–43. IEEE, 2011.
- [Fow04] Martin Fowler. *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional, 2004.
- [FRM94] Christos Faloutsos, Mudumbai Ranganathan, and Yannis Manolopoulos. *Fast subsequence matching in time-series databases*, volume 23. ACM, 1994.
- [FRMI14] Walid Fdhila, Stefanie Rinderle-Ma, and Conrad Indiono. Memetic algorithms for mining change logs in process choreographies. In *International Conference on Service-Oriented Computing*, pages 47–62. Springer, 2014.
- [GMOvB20] Thomas Grisold, Jan Mendling, Markus Otto, and Jan vom Brocke. Adoption, use and management of process mining in practice. *Business Process Management Journal*, 2020.
- [GR15] Wilfried Grossmann and Stefanie Rinderle-Ma. *Fundamentals of Business Intelligence*. Data-Centric Systems and Applications. Springer, 2015.
- [GRM15] Wilfried Grossmann and Stefanie Rinderle-Ma. *Fundamentals of Business intelligence*. Springer, 2015.
- [GS91] Rodney M Goodman and P Smyth. Rule induction using information theory. *G. Piatetsky*, 1991.
- [GSC09] Koni Grob, Julian Stocker, and Ron Colwell. Assurance of compliance within the production chain of food contact materials by good manufacturing practice and documentation - part 1: Legal background in europe and compliance challenges. *Food Control*, 20(5):476 – 482, 2009.
- [HV08] Mia Hubert and Ellen Vandervieren. An adjusted boxplot for skewed distributions. *Computational statistics & data analysis*, 52(12):5186–5201, 2008.
- [IEE16] IEEE. Ieee standard for extensible event stream (xes) for achieving interoperability in event logs and event streams. *IEEE Std 1849-2016*, pages 1–50, 2016.
- [JVKN08] Monique H Jansen-Vullers, PAM Kleingeld, and Mariska Netjes. Quantifying the performance of workflows. *Information Systems Management*, 25(4):332–343, 2008.
- [JY09] Li Junkui and Wang Yuanzhen. Early abandon to accelerate exact dynamic time warping. *International Arab Journal of Information Technology (IAJIT)*, 6(2), 2009.

- [KHP⁺19] Klaus Kammerer, Burkhard Hoppenstedt, Rüdiger Pryss, Steffen Stökler, Johannes Allgaier, and Manfred Reichert. Anomaly detections for manufacturing systems based on sensor data - insights into two challenging real-world production settings. *Sensors*, 19(24):5370, 2019.
- [KMS⁺15a] Georg Kaes, Jürgen Mangler, Florian Stertz, Ralph Vigne, and Stefanie Rinderle-Ma. Acaplan - adaptive care planning. In *BPM Demo*, pages 11–15. Springer, 2015.
- [KMS⁺15b] Georg Kaes, Jürgen Mangler, Florian Stertz, Ralph Vigne, and Stefanie Rinderle-Ma. Acaplan-adaptive care planning. 2015.
- [KR17] Georg Kaes and Stefanie Rinderle-Ma. Generating data from highly flexible and individual process settings through a game-based experimentation service. In *Datenbanksysteme für Business, Technologie und Web*, pages 331–350. Gesellschaft für Informatik, Bonn, 2017.
- [Kru14] Richard A Krueger. *Focus groups: A practical guide for applied research*. Sage publications, 2014.
- [KRVM14] Georg Kaes, Stefanie Rinderle-Ma, Ralph Vigne, and Juergen Mangler. Flexibility requirements in real-world process scenarios and prototypical realization in the care domain. In *OTM Industry Case Studies Program*, pages 55–64. Springer, 2014.
- [KSSI20] Marc Kerremans, Samantha Searle, Tushar Srivastava, and Kimihiko Iijima. Market guide for process mining, 2020.
- [KT05] Hiroko Kato and KT Tan. 2d barcodes for mobile phones. In *Mobile Technology, Applications and Systems, 2005 2nd International Conference on*, pages 8–pp. IET, 2005.
- [LFvdA13] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Discovering block-structured process models from event logs - a constructive approach. In José-Manuel Colom and Jörg Desel, editors, *Application and Theory of Petri Nets and Concurrency*, pages 311–329, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [LMM⁺15] Linh Thao Ly, Fabrizio Maria Maggi, Marco Montali, Stefanie Rinderle-Ma, and Wil M. P. van der Aalst. Compliance monitoring in business processes: Functionalities, application, and tool-support. *Information systems*, 54:209–234, 2015.
- [LR07] Richard Lenz and Manfred Reichert. IT support for healthcare processes - premises, challenges, perspectives. *Data & Knowledge Engineering*, 61(1):39–58, 2007.
- [LSvdA19] Sander JJ Leemans, Anja F Syring, and Wil MP van der Aalst. Earth movers’ stochastic conformance checking. In *International Conference on Business Process Management*, pages 127–143. Springer, 2019.

- [MBCS13] Fabrizio Maria Maggi, Andrea Burattin, Marta Cimitile, and Alessandro Sperduti. Online process discovery to detect concept drifts in ltl-based declarative process models. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, pages 94–111. Springer, 2013.
- [MdLRvdA16a] Felix Mannhardt, Massimiliano de Leoni, Hajo A. Reijers, and Wil M. P. van der Aalst. Balanced multi-perspective checking of process conformance. *Computing*, 98(4):407–437, 2016.
- [MDLRVDA16b] Felix Mannhardt, Massimiliano De Leoni, Hajo A Reijers, and Wil MP Van Der Aalst. Decision mining revisited-discovering overlapping rules. In *International conference on advanced information systems engineering*, pages 377–392. Springer, 2016.
- [MDRO17] Abderrahmane Maaradji, Marlon Dumas, Marcello La Rosa, and Alireza Ostovar. Detecting sudden and gradual drifts in business processes from execution traces. *IEEE Trans. Knowl. Data Eng.*, 29(10):2140–2154, 2017.
- [MI02] Yukio Matsumoto and K Ishituka. Ruby programming language, 2002.
- [MPO⁺19] Luca Mottola, Gian Pietro Picco, Felix Jonathan Oppermann, Joakim Eriksson, Niclas Finne, Harald Fuchs, Andrea Gaglione, Stamatis Karnouskos, Patricio Moreno Montero, Nina Oertel, Kay Römer, Patrik Spiß, Stefano Tranquillini, and Thiemo Voigt. makesense: Simplifying the integration of wireless sensor networks into business processes. *IEEE Transactions on Software Engineering*, 45(6):576–596, 2019.
- [MRM14] Jürgen Mangler and Stefanie Rinderle-Ma. Cpee - cloud process execution engine. In *Int'l Conference on Business Process Management*, CEUR-WS.org. IEEE, 2014.
- [MWVdAvdB02] Laura Maruster, AJMM Ton Weijters, WMP Wil Van der Aalst, and Antal van den Bosch. Process mining: Discovering direct successors in process logs. In *International Conference on Discovery Science*, pages 364–373. Springer, 2002.
- [OMG13] OMG. Business Process Model and Notation (BPMN), Version 2.0.2, December 2013.
- [Pet81] James L Peterson. Petri net theory and the modeling of systems. 1981.
- [PKG11] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.
- [PLCR19] Roberto Posenato, Andreas Lanz, Carlo Combi, and Manfred Reichert. Managing time-awareness in modularized processes. *Software and Systems Modeling*, 18(2):1135–1154, 2019.

- [PML12] Andreas Prinz, Philipp Menschner, and Jan Marco Leimeister. Electronic data capture in healthcare - nfc as easy way for self-reported health status information. *Health Policy and Technology*, 1(3):137 – 144, 2012.
- [PMRP18] Florian Pauker, Juergen Mangler, Stefanie Rinderle-Ma, and Christoph Pollak. centurio.work - modular secure manufacturing orchestration. In *BPM Industry Track*, pages 164–171, 2018.
- [PS⁺08] Eric Prud’Hommeaux, Andy Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.
- [RCM⁺12] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 262–270, 2012.
- [Rei20] Lars Reinkemeyer. *Process Mining in Action – Principles, Use Cases and Outlook*. Springer International Publishing, 2020.
- [RMRJ11] Stefanie Rinderle-Ma, Manfred Reichert, and Martin Jurisch. On utilizing web service equivalence for supporting the composition life cycle. *International Journal of Web Services Research (IJWSR)*, 8(1):41–67, 2011.
- [RRJK06] Stefanie Rinderle, Manfred Reichert, Martin Jurisch, and Ulrich Kreher. On representing, purging, and utilizing change logs in process management systems. In *Business Process Management, 4th International Conference, BPM 2006, Vienna, Austria, September 5-7, 2006, Proceedings*, pages 241–256, 2006.
- [RTG⁺20] V. Rodriguez-Fernandez, A. Trzcionkowska, A. Gonzalez-Pardo, E. Brzychczy, G. J. Nalepa, and D. Camacho. Conformance checking for time series-aware processes. *IEEE Transactions on Industrial Informatics*, pages 1–1, 2020.
- [RvdA06] Anne Rozinat and Wil MP van der Aalst. Decision mining in prom. In *International Conference on Business Process Management*, pages 420–425. Springer, 2006.
- [RVdA08] Anne Rozinat and Wil MP Van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008.
- [RW12] Manfred Reichert and Barbara Weber. *Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies*. Springer, 2012.
- [Sar72] G. Saretsky. The OEO P.C. experiment and the John Henry effect. *Phi Delta Kappan*, 53:579–581, 1972.

- [SC07] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.
- [SH12] P Shadiya and PP Abdul Haleem. Energy efficient data formatting scheme: A review and analysis on xml alternatives. *Energy*, 1(1):1–4, 2012.
- [SJE18] Stefan Schöning, Stefan Jablonski, Andreas Ermer, and Ana Paula Aires. Digital connected production: Wearable manufacturing information systems. In Christophe Debruyne, Hervé Panetto, Georg Weichhart, Peter Bollen, Ioana Ciuciu, Maria-Esther Vidal, and Robert Meersman, editors, *On the Move to Meaningful Internet Systems. OTM 2017 Workshops*, pages 56–65, Cham, 2018. Springer International Publishing.
- [SMR17] Florian Stertz, Juergen Mangler, and Stefanie Rinderle-Ma. Nfc-based task enactment for automatic documentation of treatment processes. In *Enterprise, Business-Process and Information Systems Modeling - 18th International Conference, BPMDS 2017, 22nd International Conference, EMMSAD 2017, Held at CAiSE 2017, Essen, Germany, June 12-13, 2017, Proceedings*, pages 34–48. Springer, 2017.
- [SRM18] Florian Stertz and Stefanie Rinderle-Ma. Process histories – detecting and representing concept drifts based on event streams. In *Cooperative Information Systems*, pages 318–335, 2018.
- [SRM20a] Florian Stertz, Stefanie Rinderle-Ma, and Juergen Mangler. Analyzing process concept drifts based on sensor event streams during runtime. In *to appear in Business Process Management, 18th International Conference, BPM 2020*. Springer International Publishing, 2020.
- [SRM20b] Florian Stertz, Stefanie Rinderle-Ma, and Juergen Mangler. Analyzing process concept drifts based on sensor event streams during runtime. In *Business Process Management*, pages 202–219, 2020.
- [SRMM] Florian Stertz, Stefanie Rinderle-Ma, and Juergen Mangler. Data set containing process execution log data with time sequence information for conference proceeding 2020 paper: Analyzing process concept drifts based on sensor event streams during runtime. <http://dx.doi.org/10.6084/m9.figshare.12472634>.
- [SSSA12] Stefan Schulte, Dieter Schuller, Ralf Steinmetz, and Sven Abels. Plug-and-play virtual factories. *IEEE Internet Computing*, 16(5):78–82, 2012.
- [SVdA08] Minseok Song and Wil MP Van der Aalst. Towards comprehensive support for organizational mining. *Decision support systems*, 46(1):300–317, 2008.
- [SvZvdA20] Mohammadreza Fani Sani, Sebastiaan J van Zelst, and Wil MP van der Aalst. Conformance checking approximation using subset selection and edit distance. In *International Conference on Advanced Information Systems Engineering*, pages 234–251. Springer, 2020.

- [T⁺13] R Core Team et al. R: A language and environment for statistical computing. 2013.
- [Tav18] Romain Tavenard. tslearn documentation. 2018.
- [The19] The Hackett Group. Enabling business process change, 2019.
- [Van12] Van Dongen, B.F. (Boudewijn). Bpi challenge 2012, 2012.
- [VBDA11] H. M. W. Verbeek, Joos C. A. M. Buijs, Boudewijn F. Dongen, and Wil M. P. Aalst. XES, XESame, and ProM 6. In *Information Systems Evolution*, volume 72, pages 60–75. IEEE, 2011.
- [vBM19] Jan vom Brocke and Alexander Maedche. The DSR grid: six core dimensions for effectively planning and communicating design science research projects. *Electronic Markets*, 29(3):379–385, 2019.
- [vdA11] Wil M. P. van der Aalst. *Mining Additional Perspectives*, pages 215–240. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [vdA16] Wil M. P. van der Aalst. *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.
- [VDAADM⁺11] Wil Van Der Aalst, Arya Adriansyah, Ana Karla Alves De Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter van den Brand, Ronald Brandtjen, Joos Buijs, et al. Process mining manifesto. In *International Conference on Business Process Management*, pages 169–194. Springer, 2011.
- [VdAAvD12] Wil Van der Aalst, Arya Adriansyah, and Boudewijn van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.
- [vdABvZ17] Wil MP van der Aalst, Alfredo Bolt, and Sebastiaan J van Zelst. Rapid-ProM: mine your processes and not just your data. *arXiv preprint arXiv:1703.03740*, 2017.
- [vdAdMW05] W. M. P. van der Aalst, A. K. Alves de Medeiros, and A. J. M. M. Weijters. Genetic process mining. In Gianfranco Ciardo and Philippe Darondeau, editors, *Applications and Theory of Petri Nets 2005*, pages 48–69, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [VDARS05] Wil MP Van Der Aalst, Hajo A Reijers, and Minseok Song. Discovering social networks from event logs. *Computer Supported Cooperative Work (CSCW)*, 14(6):549–593, 2005.
- [VdASS11] Wil MP Van der Aalst, M Helen Schonenberg, and Minseok Song. Time prediction based on process mining. *Information systems*, 36(2):450–475, 2011.

- [VdAWM04] Wil Van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
- [vZBH⁺17] Sebastiaan J van Zelst, Alfredo Bolt, Marwan Hassani, Boudewijn F van Dongen, and Wil MP van der Aalst. Online conformance checking: relating event streams to process models using prefix-alignments. *International Journal of Data Science and Analytics*, pages 1–16, 2017.
- [vZvDvdA18] Sebastiaan J van Zelst, Boudewijn F van Dongen, and Wil MP van der Aalst. Event stream-based process discovery using abstract representations. *Knowledge and Information Systems*, 54(2):407–435, 2018.
- [Wan06] Roy Want. An introduction to rfid technology. *Pervasive Computing*, 5:25–33, 2006.
- [Wie14] Roel J Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.
- [WK96] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.
- [WKVHMN05] Li Wei, Eamonn Keogh, Helga Van Herle, and Agenor Mafra-Neto. Atomic wedgie: efficient query filtering for streaming time series. In *Fifth IEEE International Conference on Data Mining (ICDM’05)*, pages 8–pp. IEEE, 2005.
- [WM12] Roel Wieringa and Ayşe Morali. Technical action research as a validation method in information systems design science. In *International Conference on Design Science Research in Information Systems*, pages 220–238. Springer, 2012.
- [WvdA03] A. J. M. M. Weijters and Wil M. P. van der Aalst. Rediscovering workflow models from event-based data using little thumb. *Integr. Comput. Aided Eng.*, 10(2):151–162, 2003.
- [WvDADM06] AJMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166:1–34, 2006.