Institute of Architecture of Application Systems

Universität Stuttgart
Universitätsstraße 38
70569 Stuttgart
Germany

Master's Thesis Nr. 0838-012

Combination Usage of Process Mining and Adaptive Case Management

Thatchanok Wiriyarattanakul

Course of Study:	Information Technology
Examiner:	Prof. Dr. Frank Leymann
Supervisor:	Dipl.-Inf. Falko Kötter
Commenced:	7 April 2016
Completed:	7 October 2016
CR-Classification	D.2.3, H.4.1, J.7

ABSTRACT

Process mining performs well on structured processes like BPM. In recent years, Adaptive Case Management (ACM) was introduced to support knowledge workers by giving them the permission to define processes on the fly in unpredictable situations. By doing so, processes seem to be unstructured and hard to be enhanced. The goal of this thesis is to analyze how process mining can improve and benefit unstructured processes and to develop a prototype for a potential application. The scenario analysis focuses on (1) supporting knowledge workers with process mining, (2) transiting from unstructured to structured processes through process mining, and (3) compliance regulations in unstructured processes through process mining. One scenario is selected based on the analysis for implementing a prototype. Due to the importance of compliance regulations and rare researches on compliance regulations in unstructured processes, the scenario (3) is selected and an approach of compliance checking for unstructured processes using process mining is proposed. The prototype of the approach leverages process mining to discover hidden structured in unstructured processes and implements compliance checking functionalities consisting of graphical rule definition, rule creation and rule checking on a log in Oryx platform. The prototype visualizes violating paths on the process model and reports all violating process instances. The evaluation proves that the prototype is indeed capable of compliance checking with a large real-life data log.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my master thesis advisor, Falko Kötter of Fraunhofer IAO, for his guidance, advices and support throughout the writing of my thesis. His office's door was always open whenever I encountered a difficulty or had a question about my research or writing. He allowed me the freedom to do this thesis on my own, but consistently steered me into the right direction to achieve the thesis goal. I would also like to express my appreciation to Sebastian Wagner, who advised me throughout the writing of my thesis and who was responsible for my thesis registration until submission processes. Also, a special warm thank you to Monica Kochanowski, who inspired this thesis topic and organized thesis meetings. Moreover, I would like to gratefully thank Prof. Frank Leymann, who was my thesis examiner as well as a brilliant professor during my master program at the University of Stuttgart. Finally, my deepest thanks go to my family and my loved ones for their unconditional support throughout my studies and my life. Without them, I would not have reached this point.

Content

Abstract i

Acknowledgements ii

List of Figures vi

List of Tables viii

1 Introduction 1

1.1 Research Scope 2

1.2 Research Methodology 3

1.3 Thesis Structure 4

2 Preliminaries 5

2.1 Process Mining 5

2.1.1 Process Discovery 8

2.1.2 Conformance Checking 9

2.1.3 Model Enhancement 10

2.2 Adaptive Case Management 12

2.2.1 Overview on Adaptive Case Management 12

2.2.2 ACM and BPM 14

2.3 Process Mining and Adaptive Case Management 17

2.3.1 Using Process Mining with Adaptive Case Management 17

2.3.2 Preparing ACM for Process Mining 18

3 Related Work and Analysis 21

3.1 Scenario 1: Process mining for supporting knowledge workers in ACM 21

3.2 Scenario 2: Process Mining for Transition from Unstructured to Structured process 25

3.3 Scenario 3: Process Mining for Compliance Regulations in Unstructured Processes 30

3.4 Conclusion and Selection of Scenario 32

4 Architecture Design 35

4.1 Architecture Overview 35

4.2 Architecture Details 36

5 Methods and Technologies 39

5.1 Process Mining 39

5.1.1 Analysis of Process Mining Techniques 40

5.1.2 Selection of Process Mining Techniques 41

5.2 Rule Definition and Rule Checking 42

5.2.1 Meaning of Business Rule 42

5.2.2 Platform Analysis for Rule Definition and Rule Checking 43

5.2.3 Selection of Rule Definition and Rule Checking Platform 47

5.3 Compliance Checking 47

5.3.1 Analysis of Compliance Checking Techniques 48

5.3.2 Selection of Compliance Checking Techniques 50

5.4 Conclusion of Methods and Technologies 51

6 Implementation 53

6.1 Event Log Preparation 54

6.2 Performing Process Mining 55

6.3 Rules Definition 56

6.3.1 PROM to Oryx converter 56

6.3.2 Graphical Rule Definition 59

6.4 Rule Creation 61

6.5 Rule Checking on Log 63

6.6 Result Visualization 66

6.6.1 Graphical Result 66

6.6.2 Information Table Result 68

7 Evaluation 69

7.1 Source of Real World Dataset 69

7.2 Log Preparation 70

7.3 Prototype Evaluation 71

7.3.1 Process Mining in PROM 71

7.3.2 Oryx Process Editor 75

8 Conclusions 89

8.1 Summary of Contributions 89

8.2 Limitations 90

8.3 Future Work 91

9 References 93

LIST OF FIGURES

Figure 1.1 The research scope is the intersection of process mining and adaptive case management 2

Figure 1.2 The research methodology 3

Figure 2.1 Three basic types of process mining: conformance checking, process discovery and model enhancement [9] 6

Figure 2.2 Common structure of event logs [2] 7

Figure 2.3 Overview of Process Discovery....8

Figure 2.4 Overview of Conformance checking [2]....10

Figure 2.5 The three basic types of process mining in terms of input and output [10]. 11

Figure 2.6 ACM characteristics [24]....13

Figure 2.7 BPM and ACM approaches [6] 15

Figure 2.8 Spectrum of Process Functionality [33] 16

Figure 3.1 A spaghetti or unstructured process is transformed to a simplified process model [37]....26

Figure 3.2 Discovery of process model from content-based log....27

Figure 3.3 The relationship between the three scenarios 33

Figure 3.4 The selected scenario illustrated in the overall approach for the prototype solution....34

Figure 4.1 Architecure overview of compliance checking in unstructured process using process mining....36

Figure 5.1 Rule formulating in different roles (adapted from [52])....43

Figure 5.2 Oryx interface for BPMN modeling. Drag and drop elements are on the left side and the properties for each element are on the right side [53]....45

Figure 5.3 Oryx architecture [53]....45

Figure 5.4 Interface of the Oryx extended by Fraunhofer IAO 46

Figure 5.5 Compliance Management Process [55]....48

Figure 5.6 Types of compliance checking (adapted from [58] , [59])....49

Figure 5.7 Overview of Methods and Technologies employing in the compliance checking in unstructured process using process mining....52

Figure 6.1 The detail architecture detail of compliance checking in unstructured process using process mining.......53

Figure 6.2 Results of process discovery produced by Fuzzy miner (left) and Heuristic miner (right) 55

Figure 6.3 Comparison of PROM and Oryx BPMN structure and mapping fields between the two structures.......58

Figure 6.4 Importing BPMN generated by PROM to Oryx process editor.......59

Figure 6.5 PROM discovered process model displayed in Oryx process editor 59

Figure 6.6 Rule definition icons consists of Correct pattern, Incorrect pattern, and Indirect flow 60

Figure 6.7 The graphical rule definition in Oryx process editor and the rules summary panel 60

Figure 6.8 Implementation of rule creation in Oryx process editor 63

Figure 6.9 Oryx Rule checking on log using Esper Complex Event Processing 64

Figure 6.10 Implementation of rule checking on log in Oryx process editor 65

Figure 6.11 Rule checking in Oryx process editor for multiple-ends processes.......66

Figure 6.12 Graphical result of rule checking on log displayed in Oryx canvas.......67

Figure 6.13 The table result of rule checking on log in Oryx process editor.......68

Figure 7.1 Log preparation 70

Figure 7.2 Log Summary generated by PROM.......71

Figure 7.3 The discovered process model by PROM.......73

Figure 7.4 The imported process model to Oryx process editor 74

Figure 7.5 The adjusted ARPOS process model.......76

Figure 7.6 ARPOS process model with graphical rule definiton.......77

Figure 7.7 Rule definition summary.......78

Figure 7.8 The modified rule checking on log panel.......81

Figure 7.9 The information table result of rule checking on ARPOS log with 27,000 executed tasks (left) and 188,850 executed tasks (the whole log over one year) (right) 82

Figure 7.10 The graphical result of rule checking on ARPOS log 83

Figure 7.11 Example of a process instance which violates rule R12 (left) and R13 (right) 86

LIST OF TABLES

Table 2.1 Key differences between BPM and ACM (adapted form [27], [28]) 15
Table 3.1 Overview of guidance approaches for knowledge workers.............................24
Table 3.2 Overview of transition from unstructured to structured process approaches 29
Table 6.1 The comparison of definitions between BPMN of PROM and Oryx 56
Table 6.2 Rule creation of graphical rules to Esper rules...62
Table 6.3 Rule checking on log result explanations..67
Table 7.1 Rule statements interpreted from the defined graphical rules 79
Table 7.2 Conclusion of defined rules, descriptions and rule checking on log result 84

1 Introduction

Process mining aims to exploit the information logs of business processes executions. The idea to apply process mining in the context of workflow management systems has been introduced since the '90s [1]. Process mining provides insight information about end-to-end processes in organizations. For this reason, process mining methods are now used in most phases of the BPM lifecycle, from discovery through monitoring [2]. A various number of applications benefit from applying process mining techniques such as constructing a process model from an event log, process improvement and conformance checking to identify deviations [3].

Business Process Management (BPM) is used to support operational activities in organizations. BPM aims to formulate a static workflow to achieve cost-effective through repeatability and automation. All the possible paths or control flows can be predicted in advance and automated at runtime [4]. However, in unpredictable situations BPM faces a flexibility issue due to its inability to predict all possible activities beforehand. Therefore, Adaptive Case Management (ACM) is introduced in such unpredictable situations. ACM aims to minimize this BPM drawback by giving flexible permissions to workers. The workers can perform their tasks with their special expertise [5] and define business process on the fly in unpredictable situations [6].

In ACM, workers are given the freedom to govern business processes at runtime. Workers decide actions depending on the current situation. Therefore, processes seem to be unstructured. In this master thesis, the focus is on the analysis of how process mining benefits unstructured processes such ACM is focused. The scenario analysis focuses on (1) supporting knowledge workers with process mining, (2) transiting from unstructured to structured processes through process mining and (3) compliance regulations in unstructured processes through process mining. As a result, process mining not only benefits structured processes but also ACM can benefit from process mining, even though ACM is unpredictable and unstructured processes [7].

After the analysis of the three scenarios, one is implemented as a prototype. Due to the importance of compliance regulations in organizations and reduced research work performed on compliance regulations in unstructured processes, scenario (3) is selected and an approach of compliance checking for unstructured processes using process mining is proposed in this master thesis. As proof of concept, PROM – a process mining tool and Oryx – a process modeling editor are employed. PROM is used for discovering a structured process from an unstructured process. Oryx platform is used for implementing new compliance checking functionalities consisting of graphical rule definition, rule creation and rule checking on log. Furthermore, the approach is evaluated with a real-life data log to verify correctness and applicability to real world situations.

1.1 Research Scope

The objective of this master thesis is to explore the potential of using together two different disciplines, one being process mining and the other being adaptive case management. Process mining aims to extract what actually happened in organization from historical data or an event log. Adaptive case management intends to support skilled workers in unpredictable situations.

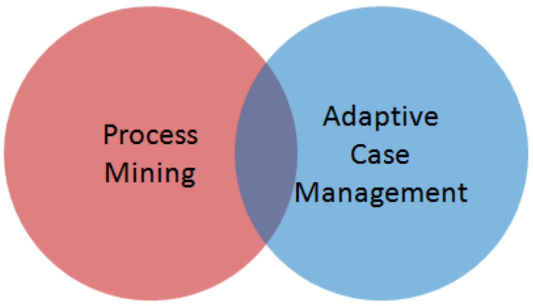


Figure 1.1 The research scope is the intersection of process mining and adaptive case management

Figure 1.1 illustrates the scope of the research which is the intersection of process mining and adaptive case management. A various number of applications benefit from applying process mining techniques such as constructing a process model from an event log and identifying deviations. Adaptive case management is used in exceptional handling in uncertain environment. In this master thesis, identifying the intersections between process mining and adaptive case management is investigated i.e. analyzing how process mining can benefit adaptive case management.

1.2 Research Methodology

The research methodology applied in this master thesis is illustrated in Figure 1.2. The method consists of scope setting, analyzing defined scenarios, selecting a scenario, designing the prototype architecture, identifying methods and technologies, implementing and evaluating the prototype solution with real world data.

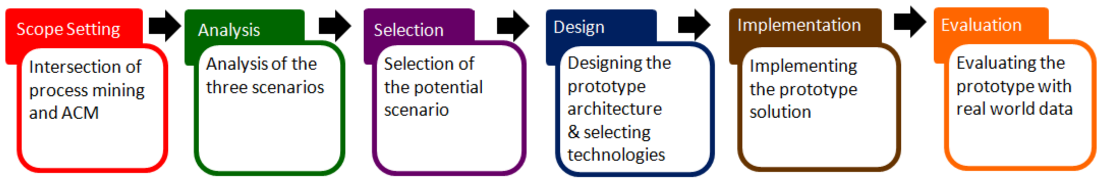

Scope Setting
Intersection of process mining and ACM
Analysis
Analysis of the three scenarios
Selection
Selection of the potential scenario
Design
Designing the prototype architecture & selecting technologies
Implementation
Implementing the prototype solution
Evaluation
Evaluating the prototype with real world data

Figure 1.2 The research methodology

The research scope is defined to leverage the power of process mining in combination with ACM (Section 1.1). Then, the current process mining and ACM approaches with their similarities and differences are analyzed in respect to these three scenarios (Chapter 3):

- Scenario 1: Could process mining be a useful extension for adaptive case management solution for knowledge workers?
- Scenario 2: In an unstructured process, could process mining be used to ensure compliance for certain regulations?
- Scenario 3: Can process mining be used to enable a transition from unstructured to partly structured processes?

Afterwards, the potential combined approach is selected based on the analysis of the three scenarios (Section 3.4). Then, the selected scenario is designed into the prototype architecture (Chapter 4). Subsequently, the methods and technologies are discussed to identify the best suitable techniques for the approach (Chapter 5). Finally, the prototype solution is implemented (Chapter 6) and evaluated with real world log data (Chapter 7).

1.3 Thesis Structure

The thesis is structured as follows:

Background concepts and related definitions used throughout this thesis are introduced. The concepts and definitions cover the aspects of process mining, adaptive case management and business process management (Chapter 2).

Related work is analyzed in the scope of process mining and adaptive case management in unstructured process applications. At the end of this chapter, the scenario is selected in order to implement a prototype solution (Chapter 3).

Architecture overview of the prototype solution is demonstrated and the detail of each component will be also explained (Chapter 4).

Methods and technologies of each component are analyzed and the suitable techniques for the approach is selected (Chapter 5).

The implementation of the prototype solution for the proposed approach and the challenges encountered during implementation are discussed (Chapter 6).

The implementation is evaluated and experimental results are shown. Correctness of the approach is verified using a real-life event log (Chapter 7).

A summary of the entire research and a discussion about the limitations and future work are presented (Chapter8).

2 Preliminaries

In the previous chapter, the research goal, scope and methodology of this master thesis are demonstrated. This chapter discusses an overview of concepts and related definitions on which this master thesis relies.

2.1 Process Mining

Process mining is the analysis of processes using executions logs [3]. The idea to apply process mining in the context of workflow management systems has been introduced since the '90s [1]. Process mining provides insight information about end-to-end processes in organizations. For this reason, process mining methods are now used in most of the phases of the BPM lifecycle, not only in the design phase but in the enactment, monitoring and adjustment phases as well [2]. An obvious example is the use of process mining in the diagnosis phase. In diagnosis phase, process mining is used to identify opportunities for process improvement and to provide ideas for redesign.

Van der Aalst has given the definition of process mining as:

> *“**Process mining** aims to discover, monitor and improve real processes by extracting knowledge from event logs readily available in today’s information system”... Van der Aalst 2011* [2]

Process mining discovers knowledge from event logs and graphically represents a business process with a process model. The discovered process model reflects to reality by describing the dependencies between executed activities [8].

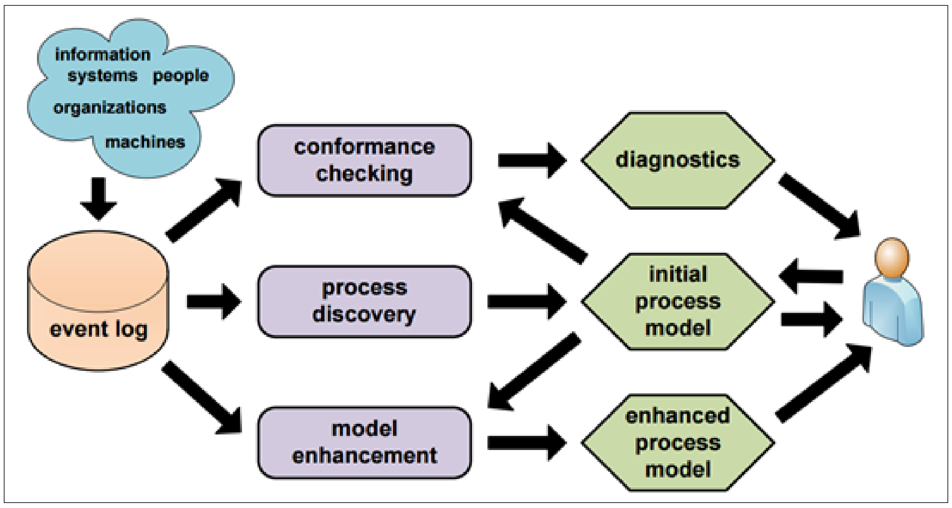


Figure 2.1 Three basic types of process mining: conformance checking, process discovery and model enhancement [9]

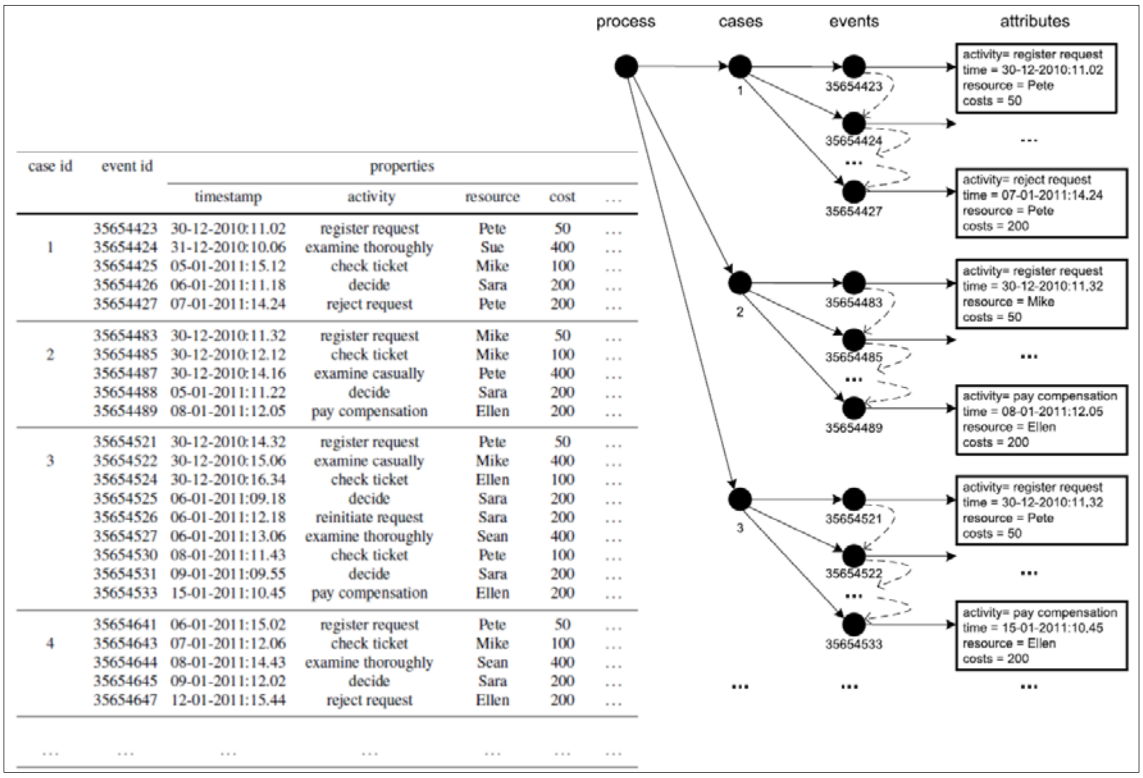
Figure 2.1, there are three basic types in process mining consisting of process discovery, conformance checking and model enhancement. *Process discovery* technique discovers a model by only using an event log as input. This results in a so-called initial process model. The discovered process model can be used in both conformance checking and model enhancement. *Conformance checking* techniques diagnoses the actual behavior (event log) with modeled behavior (initial process model) to identify deviations between the log and process model. The diagnostic information from conformance checking may be used in *Model enhancement.* The idea of model enhancement is to improve or extend the initial process model by considering diagnostic information [9].

There are some common misunderstandings about the characteristics of process mining which are listed as follows [10].

- *"Process mining is not limited to control-flow discovery".* Process mining is often seen as an approach for process discovery. However, process discovery is only one type of three basic types in process mining (See Figure 2.1). Furthermore, process mining can also construct not only control-flow perspective but also time and organizational perspective.

- *"Process mining is not just a specific type of data mining".* Data mining focuses on data-centric. However, process mining combines the strength of data-driven and process-centric so that performance and conformance aspects can be investigated.

- *"Process mining is not limited to offline analysis".* A metaphor of process mining is "Post mortem" due to process mining extracting knowledge from historical data. However, process mining can be used on a running case at runtime. For example predictions and recommendations systems [11], [12], [13], [14] and [15].



case id	event id	properties				
		timestamp	activity	resource	cost	...
	35654423	30-12-2010:11.02	register request	Pete	50	...
1	35654424	31-12-2010:10.06	examine thoroughly	Sue	400	...
	35654425	05-01-2011:15.12	check ticket	Mike	100	...
	35654426	06-01-2011:11.18	decide	Sara	200	...
	35654427	07-01-2011:14.24	reject request	Pete	200	...
	35654483	30-12-2010:11.32	register request	Mike	50	...
2	35654485	30-12-2010:12.12	check ticket	Mike	100	...
	35654487	30-12-2010:14.16	examine casually	Pete	400	...
	35654488	05-01-2011:11.22	decide	Sara	200	...
	35654489	08-01-2011:12.05	pay compensation	Ellen	200	...
	35654521	30-12-2010:14.32	register request	Pete	50	...
3	35654522	30-12-2010:15.06	examine casually	Mike	400	...
	35654524	30-12-2010:16.34	check ticket	Ellen	100	...
	35654525	06-01-2011:09.18	decide	Sara	200	...
	35654526	06-01-2011:12.18	reinitiate request	Sara	200	...
	35654527	06-01-2011:13.06	examine thoroughly	Sean	400	...
	35654530	08-01-2011:11.43	check ticket	Pete	100	...
	35654531	09-01-2011:09.55	decide	Sara	200	...
	35654533	15-01-2011:10.45	pay compensation	Ellen	200	...
	35654641	06-01-2011:15.02	register request	Pete	50	...
4	35654643	07-01-2011:12.06	check ticket	Mike	100	...
	35654644	08-01-2011:14.43	examine thoroughly	Sean	400	...
	35654645	09-01-2011:12.02	decide	Sara	200	...
	35654647	12-01-2011:15.44	reject request	Ellen	200	...
...	...	...	...	...	...	...

Figure 2.2 Common structure of event logs [2]

An event log is the starting point of process mining as shown in Figure 2.2. An event log is recorded by a software system. A common structure of an event log is illustrated in Figure 2.2. A process consists of cases or completed process instances. Each case is made of a sequence of events, called a trace. An event can have any kind of additional attributes (timestamps, cost, resource, etc.) depending on the purposes of an organization. These additional attributes are important for some analysis, for example bottlenecks which slow down the process flow [3].

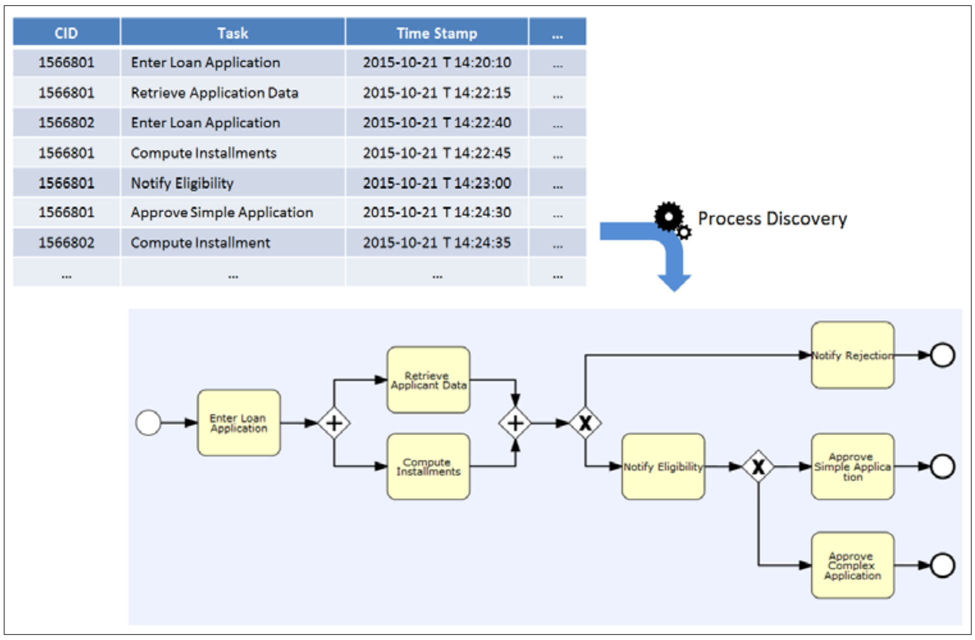
The formats of event logs may be various depending on information systems or purposes. However, the important point is the quality of event logs. The reason is that the result of process mining is heavily affected by the input. Therefore, event logs should be treated as first-class citizens in the information systems [10].

However, IEEE Task Force on Process Mining has selected XES (Extensible Event Stream)[1] as the standard format for event logs. In addition, XES format is supported by popular process mining tools such as ProM[2] and Disco[3].

In the next section, process mining types including process discovery, conformance checking and model enhancement are discussed in more details. Also, the applications and the examples of algorithms applied for each type are presented.

2.1.1 Process Discovery

Process discovery techniques transform an event log into a process model. The output process model sufficiently describes observed behaviors without using any additional information, i.e. automatically constructs a process model based only on an event log [2]. Figure 2.3 illustrates the overview of process discovery. The log entries produced from process executions are fetched into process discovery. Process discovery analyzes the logs and constructs an output process model out of the logs.



CID	Task	Time Stamp	...
1566801	Enter Loan Application	2015-10-21 T 14:20:10	...
1566801	Retrieve Application Data	2015-10-21 T 14:22:15	...
1566802	Enter Loan Application	2015-10-21 T 14:22:40	...
1566801	Compute Installments	2015-10-21 T 14:22:45	...
1566801	Notify Eligibility	2015-10-21 T 14:23:00	...
1566801	Approve Simple Application	2015-10-21 T 14:24:30	...
1566802	Compute Installment	2015-10-21 T 14:24:35	...
...	...	...	...

Figure 2.3 Overview of Process Discovery

1 http://www.xes-standard.org/
2 http://www.promtools.org/prom6/
3 http://fluxicon.com/disco/

Information systems have been designed to enforce procedures in an organization. However, the actual processes may be different in reality. Therefore, the discovery of the actual behaviors from event data could give significant insider information to organizations for future improvement. For example, process improvement and WFM/BPM configurations (discovering a process model and using it as a template for configuration) [3].

Examples of Process Discovery Algorithms

Alpha-algorithm [16], it scans the event log for particular patterns. The alpha-algorithm is simple but efficient. However, it still has problems in dealing with complicated routing constructs and noises.

Region-based approaches, which include State-based regions [17] and language-based regions [18], are able to discover more complex control flows. The basic idea is to discover places and add places that do not exclude any of the behavior seen in the event log.

Heuristic mining [19] and fuzzy mining [20] are notable algorithms for handling noises and incompleteness. Noises refer to abnormal behaviors. Incompleteness represents having insufficient data for use in process mining [8]. Both algorithms discover a process model based on occurring frequency of activities and number of times of one activity followed by another activity.

2.1.2 Conformance Checking

An event log is compared with the process model of the same process, i.e. checking if the reality conforms to the process model and vice versa. The model may have been predefined or discovered by process discovery.

Figure 2.4 illustrates a comparison of observed behavior with modeled behavior. *Global conformance measures* show the overall result of conformance checking. *Local diagnostics* mark contradictions between model and event log by highlighting on nodes [2].

Conformance checking techniques consider events in the log as activities in the model, e.g., events are mapped to transition firings in the Petri net. By this way, the observed behaviors in the event log can be compared to the modeled behavior. There are various applications of conformance checking. For instance identifying deviations, evaluating the quality of a discovered process model, auditing purposes and model enhancement.

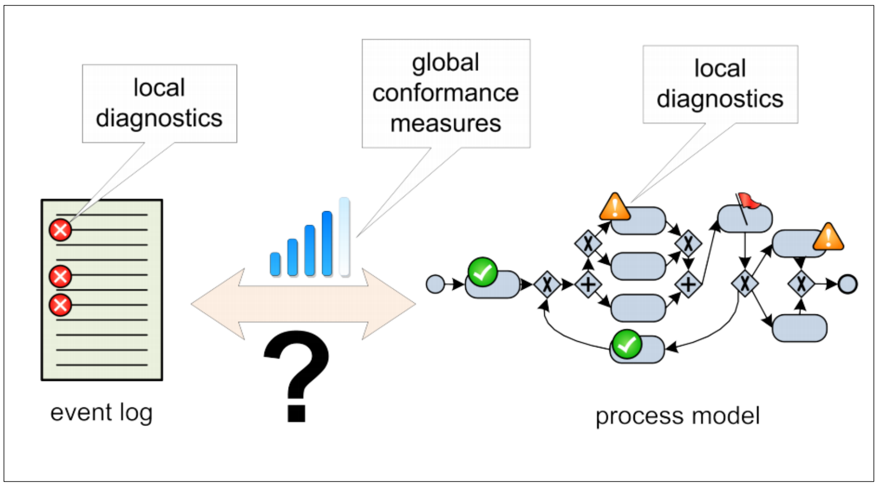


Figure 2.4 Overview of Conformance checking [2]

Examples of Conformance Checking Algorithms

Conformance checking techniques can be concluded in three categories [2]. The first approach is *Comparing Footprint*. Footprint is a matrix in both the model and the event which shows dependencies. This allows log-to-model comparison whether log and model fit on the sequence of activities. Moreover, this method can be used in log-to-log and model-to-model comparison.

The second approach, *Token Replays* the event log on the model. This normally is used to compute *fitness* quality criteria. Fitness measures "the proportion of behavior in the event log possible according to the model". It replays the trace on the model and records all situations where a transition is forced to fire without being enabled, i.e. all missing tokens are counted and also the remaining token at the end.

The third approach is *Alignment* and it is the most advanced approach. It computes an optimal alignment between each trace in the log and the most similar behavior in the model. A perfect alignment is one where all the moves of the trace in the event log can be followed by moves of the model.

2.1.3 Model Enhancement

Model enhancement also considers an event log and a process model as inputs. This means, it is possible to improve an existing process model by looking in the past. Common aspects in model enhancement are time and cost. After discovering a process model from an event log, the discovered process model can be used to analyze for performance indicators, for example average throughput time and costs for improving or reengineering the process. The bottleneck problem can be identified by analyzing waiting times between activities. After identifying the cause of bottlenecks, the process model can be enhanced at the right places. Enhancing resource performance is one important aspect. A social network in a workplace can be constructed by process discovery. It can give an idea of work collaborations and balance workload to improve resource performance. There is no restricting procedure how a process model can be enhanced. It depends on what problems an organization discovers and how an organization wants to improve.

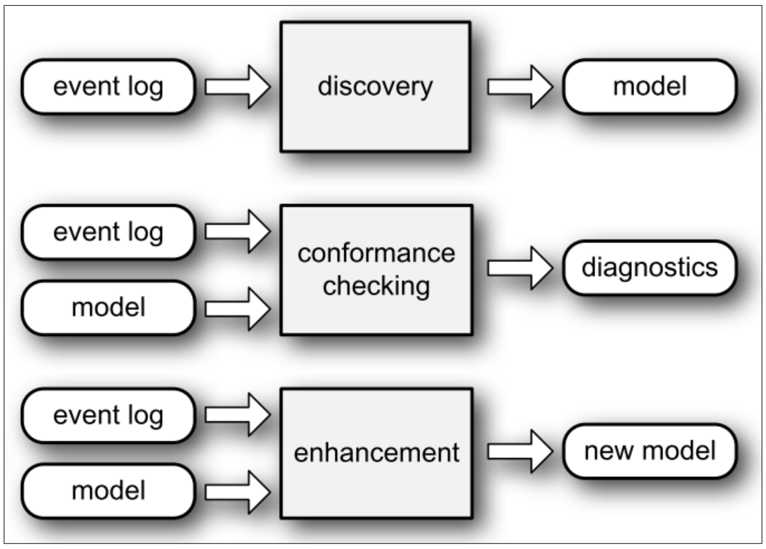


Figure 2.5 The three basic types of process mining in terms of input and output [10].

In conclusion, process mining is a very useful technique in process analysis. It can be used in various applications such as discovering a process model, auditing and improving processes. The input and output of each process mining type can be concluded in Figure 2.5. The three types of process mining need an event log as an input. It is sufficient to provide only an event log for process discovery, while both an event log and its process model are need in conformance checking and model enhancement. The outputs from each type subject to respond in different objectives. An output model constructed from process discovery aims to reveal what actually has been done. The diagnostics analyzed by conformance checking tells what has been done wrong. The consideration of what has been done in the past can be used in model enhancement to produce a new improved process model.

2.2 Adaptive Case Management

Technology solutions in the past decades have significantly changed the way of working in organizations. The routine and less skilled works have been replaced and automated by machines or software systems [21]. As the result, workers can spend less time on routine works, and dedicate their time on more complex tasks which requires a lot of thinking [22]. Therefore, there is a need for a new approach to support those people who think for living i.e. knowledge workers. That is a new paradigm called adaptive case management.

"The most valuable asset of the 20th-century company was its production equipment. The most valuable asset of a 21st-century institution will be its knowledge workers and their productivity." ...Drucker 1999 [36]

2.2.1 Overview on Adaptive Case Management

The main goal of adaptive case management (ACM) is to achieve a high amount of flexibility during process executions. ACM is used to support knowledge workers in unpredictable situations. Knowledge workers perform their works in a flexible manner with the principle of planning-by-doing. The works performed by knowledge workers are the so-called knowledge works which often cannot be predefined beforehand, therefore they have a very low degree of repeatability. Due to the detailed differences from case to case, actions must be decided by considering the current situations. With ACM, the productivity of knowledge workers can be improved because knowledge workers are given a permission to use their expertise to make a smart solution which is not just an automated decision for them [23].

*"**Adaptive Case Management** is a system that is able to support decision making and data capture while providing the freedom for knowledge workers to apply their own understanding and subject matter expertise to respond to unique or changing circumstances within the business environment" ...Swenson 2010* [25]

The key characteristics of adaptive case management [24] are given in Figure 2.6.

- Adaption: This is the ability which requires adapting to changes at current situations, in other words being flexible during runtime. With every execution, process templates can be continuously improved with feedback from users.

- Organization: Goal and data are important requirements in ACM. While, data often changes in uncertain situation, goals remain the same. Therefore, knowledge workers need to stick with the goals and handle cases by considering data which is currently being available.

- Case handling: In complex cases, knowledge workers require collaboration with several roles to solve them. Collaboration requires transparency which means it is observable and shareable to others. Resources should be integrated, including tools, and information should be gathered and available for knowledge workers.

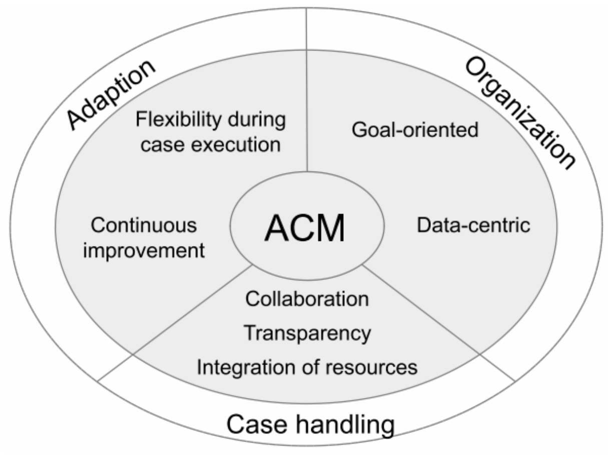


Figure 2.6 ACM characteristics [24]

The new concept of case management also comes with several synonyms e.g. Case handling, Adaptive case management, Dynamic case management and Advanced case management. The idea of having cases and dealing with them is not new, but the way these cases, knowledge works and knowledge workers are handled and supported is new [25].

A large number of tool vendors claim to support ACM methodology such as OpenText, Isis Papyrus, Oracle, and IBM. However, process models created from different tools may not be exchangeable without conversion [26]. To address this issue, Object Management Group (OMG) has published the case management standard Case

Management Model and Notation 1.0 (CMMN). CMMN is a standard to define elements in Case Management products. It is working in the same way as Business Process Model and Notation (BPMN) standardizing models in business process management (BPM) [27].

Challenges in ACM

Due to knowledge worker free to choose, the processes seem to be weakly structured. The challenge is how can ACM improve its processes in unstructured conditions. As well as, the challenge of balancing between structured processes for repetitive aspects and unstructured processes to facilitate creative aspects. Additionally there is the need to store and to extract information from highly unstructured processes and make it accessible to other knowledge workers [28]. Another interesting challenge is to provide a support system to provide guidance to knowledge workers during running cases to help identify the best solution in particular or unknown situations [29].

2.2.2 ACM and BPM

Business Process Management (BPM) has become the business culture over the last decade [30]. BPM targets to automate operational activities in business processes to achieve business goals. With traditional BPM, all the possible paths and activities are defined in advance and automated at runtime [4]. At the same time, Adaptive Case Management (ACM) has been introduced to fulfill the requirement of managing processes in adaptive environments. ACM aims to support processes that are involved with knowledge workers, therefore it allows a high degree of flexibility that gives knowledge workers permission to perform their works by applying their special expertise [5].

The term Business Process Management (BPM) has been given a definition by Palmer [31].

*“**Business Process Management (BPM)** is a discipline involving any combination of modeling, automation, execution, control, measurement and optimization of business activity flows, in support of enterprise goals, spanning systems, employees, customers and partners within and beyond the enterprise boundaries.”...Palmer* [31].

The comparison of traditional BPM and ACM approaches is illustrated in Figure 2.7. The early style BPM has the controls or defined processes as the core of the system. It lets the data flow through control flows as process instances. This means the process is primary and normally static. In contrast, ACM places data in the center which is able to support the surrounding processes to make decisions whenever necessary. The data is considered as a primary in ACM. In most cases, processes may not be fully predefined, knowledge workers have to define them on the fly depending on the situation [32]. All in all, the traditional BPM focuses on the predefined process routes, on the other hand, with ACM the case itself is the main focus [23].

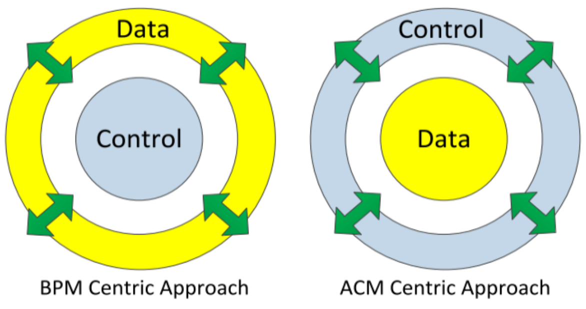


Figure 2.7 BPM and ACM approaches [6]

Table 2.1 Key differences between BPM and ACM (adapted form [27], [28])

BPM	ACM
Able to define an ordered sequence of activities	Able to define an unordered set of activities
The sequence of activities is stable and hardly changes. The process is predictable and repeatable.	The activities occur unpredictably
The process determines the events	The events determine the process
The activities are defined, can be automated and repeatable.	People determine activities and select the best solution for each case
External documents are not an essential part of the process	External documents are important to make decision

Business Situation	
Best supported with BPM: • Highly predictable and highly repeatable • Example: signing up for cell phone service: it happens thousands of times a day, and the process is essentially fixed.	Best supported with ACM: • Unpredictable and unrepeatable • Example: crime investigation, it requires following up on various clues, which are unpredictable. There are various tests and procedures to use, but they will be performed only when needed.

In practices [33], however, only few business processes are purely structured or purely unstructured. Most business functions combine structured processes (BPM) and unstructured processes (ACM) together to achieve business goals.

Traditional BPMS have been applied with excellent results to routine work, but it has been more challenging to adapt these systems to handle knowledge work. Therefore, ACM steps in to close the BPM's deficiencies loopholes and to manage unstructured segments by focusing on supporting knowledge workers rather than automation [33].

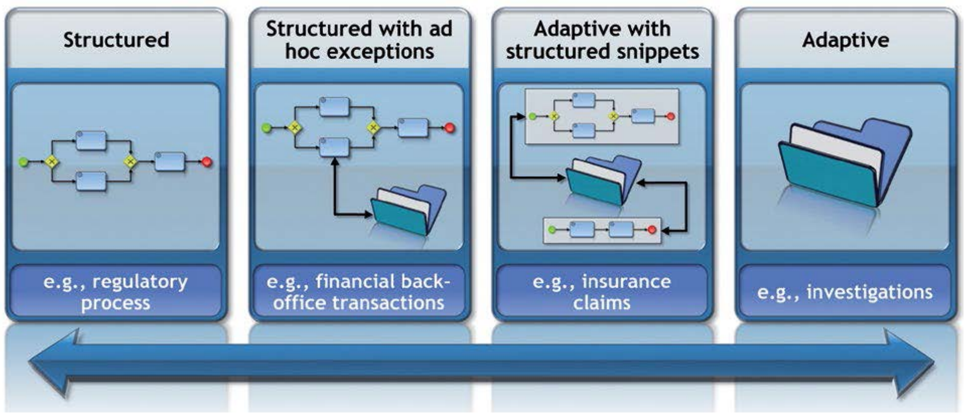

Structured
Structured with ad hoc exceptions
Adaptive with structured snippets
Adaptive
e.g., regulatory process
e.g., financial back-office transactions
e.g., insurance claims
e.g., investigations

Figure 2.8 Spectrum of Process Functionality [33]

In Figure 2.8 [33], on the left side of the spectrum are *structured processes* e.g. regulations and compliance processes which are mostly automated to achieve the goal of maximizing efficiency. In contrast, on the right side are *adaptive processes* which are unpredictable e.g. investigations. There is no benefit in modeling at all, instead knowledge works have to be performed based on each context in order to solve the problem.

Between these two extreme approaches, the left-middle is *structured with ad hoc exceptions* e.g. financial back office transactions. The structured process requires case management for exception handling. In an insurance application, for example, an application cannot be processed due to missing information and is forwarded to an exception task where a worker could decide what actions to take in order to complete the process, such as contacting the customer to request the missing information.

The right-middle is *adaptive with structured snippets* e.g. insurance claims. The ad hoc process may require some structured process parts. For example, an insurance claim process, the claim process is mainly dynamic where the claim workers decides what tasks to perform and in which order. The claim workers may request information from customers or involved parties based on each claim and consider all related information to make their final decision. However, this unstructured case management for claims has also some structured process parts in order to complete cases e.g. a claim form or other supporting documents, if they arrive via paper of fax, then they will go through a structured process such as data entry or scanning.

On the whole, ACM is not considered to replace BPM. Instead, by supporting weakly structured knowledge intensive business processes, ACM complements traditional BPM by covering a wider business process spectrum [26].

2.3 Process Mining and Adaptive Case Management

Process mining is mainly designed to discover repeatable business processes. However, increasing the efficiency of routine works leads to the enhancement of the overall processes to become ACM system. Interestingly, process mining can be used to improve ACM by discovering hidden routine processes in ACM system, helping knowledge workers in decision making and improving the case management procedures [7].

2.3.1 Using Process Mining with Adaptive Case Management

ACM aims to support knowledge workers to do their work in unpredictable situations. Therefore, designing a process model for ACM may not be worth doing it due to the diversity between cases. The process model would never end in changing and the event log produced from executing that process model would be very complex. For this reason, process mining was considered that it may not benefit to ACM because ACM is an unstructured process, while process mining is working well for structured processes [7]. However, [34] suggests that it is not necessary to define the whole process model ahead of time, but rather define the core processes and provide the right resources and tools for knowledge workers at the right time. This means there are still structured processes hiding in unstructured processes and ACM could still benefit from process mining.

[7] suggests possibilities for using process mining in ACM :

- Process mining could help to improve routine processes existing in ACM since knowledge workers still use some routine processes to accomplish their tasks.
- Hidden structured processes in ACM could be obtained by process mining. Also, the discovered patterns could develop to best practices or templates and be shared with other knowledge workers.
- Process mining could help in knowledge workers' decision making by capturing their behaviors and suggesting guidelines based on the actions performed in past similar cases.
- Process mining could be used to generate relationships between resources such as social networks and communication flows among knowledge workers.

To summarize, process mining is a highlight technique for understanding and improving not only for structured processes but also ACM can benefit from process mining even though ACM is unpredictable and unstructured processes.

2.3.2 Preparing ACM for Process Mining

ACM can benefit from process mining, however ACM must provide process mining the sufficient information. [7] suggests some guidelines for storing information in event logs as follows:

- Consistency in event logs: what knowledge workers have performed has to be recorded in event logs in a consistent manner so that process mining can produce accurately a process model from the event logs.
- Including information about resources: it would be beneficial to show social networking in order to see the communication between knowledge workers. The data tags of interactions such as requesting, waiting, reporting or handing can be added in the event logs to see complete flows between resources.

- Template column: this is useful information used for tracking and for future improvements. For example, which template is used the most in certain situations or rarely used?

Eventually, there is not a strict rule for adding information in event logs. It depends on what organization wants to find out about processes, knowledge works, and knowledge workers in adaptive case management system.

Finally, there are various ways to leverage the power of process mining in combination with the adaptive case management discipline. The focus of this work is the analysis of the combinations in the following three scenarios.

Focused scenarios:

- Scenario 1: Could process mining be a useful extension for adaptive case management solution for knowledge workers?
- Scenario 2: In an unstructured process, could process mining be used to ensure compliance for certain regulations?
- Scenario 3: Can process mining be used to enable a transition from unstructured to partly structured processes?

In the next chapter, the related work and important terms of each scenario will be discussed.

3 Related Work and Analysis

In chapter 2, the related backgrounds are discussed including process mining, adaptive case management (ACM), business process management (BPM) and combined usages of process mining and adaptive case management.

In this chapter, related work is analyzed in the scope of process mining and adaptive case management in unstructured process applications, particularly in the three scenarios below. The important terms of each scenario will also be introduced. At the end of this chapter, one scenario based on the analysis of related work will be selected in order to implement a prototype solution.

- Process mining for supporting knowledge workers in ACM
- Process mining for transition from unstructured to structured process
- Process mining for compliance regulations in unstructured process

3.1 Scenario 1: Process mining for supporting knowledge workers in ACM

The term "knowledge worker" was first introduced by Drucker in 1959 [35]. Since then knowledge-based work has become increasingly important in businesses worldwide. Drucker also suggested that "the most valuable asset of a 21st-century institution, whether business or non-business, will be its knowledge workers and their productivity" [36].

Drucker has given the definition of knowledge worker as:

Knowledge worker *is someone who knows more about his or her job than anyone else in the organization... Drucker 1959* [35]

There are various definitions for knowledge worker. Another well-known definition was introduced by Devenport's "Think for a living".

Knowledge worker *has high degrees of expertise, education, or experience, and the primary purpose of their jobs involves the creation, distribution, or application of knowledge. Knowledge workers think for a living...Davenport 2005* [70]

Davenport was also given a definition of knowledge worker as follows:

Knowledge work *is described as "untidy... the inputs and outputs of knowledge work—ideas, interruptions, inspirations, and so on are often less tangible and discrete. There are no predetermined task sequences that, if executed, guarantee the desired outcome. Knowledge workers may operate by an intuitive feel for how to accomplish their work or through accumulated experience."...Davenport 1996* [71]

Adaptive Case Management (ACM) supports the coordination of knowledge works. The nature of knowledge work is not routine-based and unpredictable. Therefore it requires knowledge workers to make decisions depending on current context.

However, without guidance, knowledge workers face difficulties when making decisions in such adaptive environments. They may not choose the best action to perform resulting in delays and unreached goals. Therefore a supporting system is important. [11], [12], [13], [14] and [15] propose approaches which apply data and process mining for prediction or recommendation systems to support knowledge workers.

The new process mining proposal in [11] is interested on partial traces to check the last executed step and uses to predict the future of a case and to recommend next possible steps. The prediction and recommendation are based on remaining time. The remaining time discovered from past cases is taken to predict the remaining execution time of the currently running cases. Also it is used to generate the next recommended steps which lead to reduced execution duration. However, this approach is implemented only in time-based aspect, the other features e.g. cost, quality, compliance will be developed in the future.

[12] presents an approach to handling knowledge-intensive business processes with low cost and accuracy. The approach consists of process skeleton and learning of business rules. The process skeleton is the core of tasks that always has to be executed. It is quick and simple to be modeled at low cost. The modeled skeleton is deployed as a workflow but at runtime knowledge workers have the authority to add resources and subtasks on their decisions. The performed tasks are captured in an event log which is used in learning business rules process. It supports knowledge workers by recommending subtasks and resources for future executions based on context and historical user behaviors. The experiment in learning business rules has been done in the case of the supervised approach through classification while in the case of unsupervised approach with an apriori algorithm. The approach is conducted on real-life business process and shows that both learning methods provide meaningful rules even with a small training set.

The sliding windows model or stream data mining has been implemented in [13]. Historical event logs (in XES format) are transformed with the sliding window model into a predictive clustering tree. Then during running cases, the tree is used to forecast the next event by predicting the properties of future events. Their experiment on various event log contexts shows that the more the size of the sliding window expands, the more the forecasting accuracy increases, but in turn it leads to a higher model complexity and a longer learning time. The predictive capability of this approach accomplishes high forecast accuracy. This ability can be used to provide conformance checking and recommendation action later on.

A general concept is presented by [14], which is based on process mining and quantitative and qualitative data. Fuzzy Miner has been applied and it has already provided quantitative figures. To include the qualitative figures, they have equipped the algorithm with additional information e.g. minimal, maximal, cost, average duration. The prioritization function calculates priority on each edge based on quantitative and qualitative information. The sorted list with respect to calculated priority presents all possible next activities recommendations to knowledge worker. The knowledge worker is free to choose one alternative from the list or decide to do something totally different. Although the approach is very generic, the approach has been applied to real data in the area of insurance claim management.

[15] has designed and developed a prototype that provides next step recommendations and predictions. The prototype consists of four main components: event log generator, model builder, next model and predictor/recommender. The event log generator transforms logs from different sources in ACM system into XES format. The model builder uses event log to derive different models. Those models are merged into a composite model or the next model. Finally, the predictor/recommender uses the next model to give suggestions for running cases based on various data and process mining techniques. The prototype gives case predictions about remaining time, possible deadline violations and supports the given goal. It also provides recommended actions with reasons which shorten the case duration, avoid deadline transgression, support case goals and are used in similar cases. The evaluation has shown that the prototype has the capability to generate predictions and recommendations. However, the chosen data set contained fictive cases. Real world datasets may result in unreliable predictions and recommendations.

Table 3.1 Overview of guidance approaches for knowledge workers

Author	Prediction	Recommendation	Approach
[11] Aalst et al. 2010	✓	✓	Estimate remaining execution time of running case from past cases based on time aspect
[12] Witschel et al. 2012	-	✓	Partially define core process, knowledge workers are freed to add desired subtasks based on learning business rules from log files
[13] Appice et al. 2013	-	✓	Generate recommendations without having pre-defined processes with sliding windows of time intervals technique
[14] Heber et al. 2015	-	✓	Modify fuzzy miner algorithm with additional qualitative information

[15] Huber et al. 2015	✓	✓	Prototype suggests remaining time, deadline violations and supports the given goal

In conclusion, process mining techniques are not only limited to past information analysis but also benefit the present and future forecasting to support knowledge workers in such ACM system. [11] and [15] have implemented both predictions and recommendations based on process mining. While [11] take into consideration the remaining time discovered from past cases to estimate the remaining execution time of the currently running cases, the prototype by [15] gives suggestions on remaining time and also possible deadline violations and supports the given goal.

The Best next step recommendation has been described in [14], [13], and [12]. Whereas the core processes have been partially defined in [12] approach, [13] and [14] generate recommendations without having pre-defined processes. In [13], sliding windows of time intervals technique has been used to suggest the target properties of future events. Therefore the next action can be created arbitrarily. In contrast, [14] equipped fuzzy miner algorithm with additional qualitative information. All possible next activities are listed ordering by priority calculations. Although, the core processes are defined in [12], the approach gives freedom to knowledge workers to add desired subtasks based on learning business rules from log files.

3.2 Scenario 2: Process Mining for Transition from Unstructured to Structured process

A traditional workflow language aims at constructing a well-defined process and highly automated. However, in knowledge intensive settings, processes are difficult to predict beforehand, therefore adaptability during executions is required. As a result, processes become less structured. Structured process is referred to as a rigorously defined process, less complex and with high repetition frequency. On the other hand, unstructured process is not or partially predefined, adaptable, content-driven and knowledge worker involved.

The definition of a structured process as given by Devonport is as follows:

Structured Process *defined as a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs: a structure for action...Devonport 1998* [72]

The definition of a structured process as suggested by Ukelson is as follows:

Unstructured process*: every instance of the process can be different from another based on the environment, the content and the skills of the people involved. These are always human processes. These processes may have a framework or guideline driving the process, but only as a recommendation... Ukelson 2009* [73]

Executing loosely structured processes generates unstructured behaviors. After mining an unstructured log, a spaghetti-like process can be revealed. "Spaghetti processes" is a metaphor of unstructured processes. It cannot be assumed that a spaghetti-like process is wrong or that it has a problem caused by process discovery algorithms. It rather means a process model accurately reflects reality. However, spaghetti processes still have issues that are difficult to be analyzed and hard to understand due to the complexity in unstructured process models. Therefore, it is a very interesting challenge to simplify an unstructured process into a more structured one.

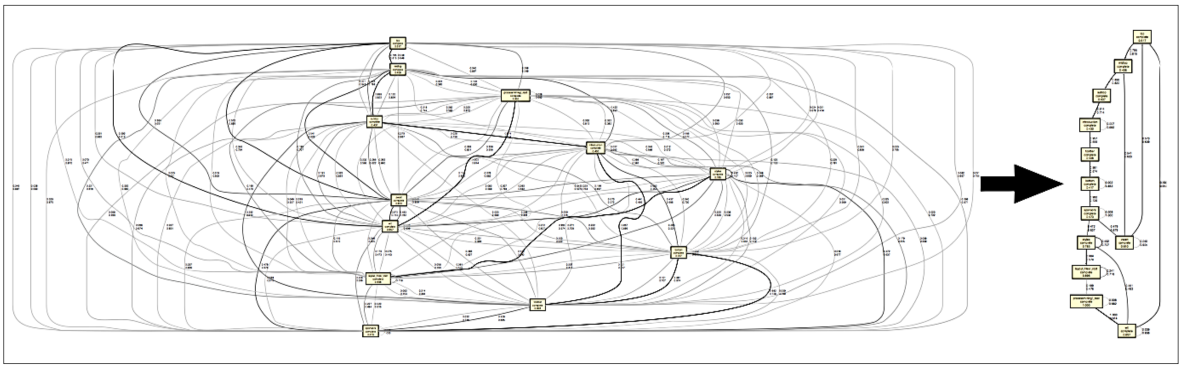


Figure 3.1 A spaghetti or unstructured process is transformed to a simplified process model [37]

Figure 3.1 illustrates the transformation of a spaghetti process model to become more comprehensible and a simpler process model. The idea of simplifying process models is an abstraction which keeps high frequent behaviors and filters out low frequent behaviors. The approaches in [38] and [39] intend to reduce complexity of process models with abstraction techniques.

[38] proposed a simplification technique for less structured "spaghetti-like" processes by fuzzy miner. The fuzzy miner has been implemented and integrated with the decision criteria "significance" and "correlation" by applying the concepts of abstraction and aggregation. The behaviors with high significance appear in the model, the less significant ones but with high correlation are aggregated in clusters, while other cases are removed. The outcome model is simplified and comprehensible. Their approach accomplishes to clean up a huge amount of confusing behaviors and "bring structure to the unstructured" which leads to process improvement.

The approach in [39] transforms raw event logs to become more abstract by subjecting to discovery of common pattern resulting in mined process model which is then less spaghetti-like. Those common behaviors are identified by the loop constructs patterns and replaced with abstract entities. This technique has been applied on a real-life log in healthcare domain and mined by heuristics miner. The results came out promising, thereby the mined model is shown more simplified, more expressive and more comprehensible. The approach can be seamlessly integrated as a pre-processing step with other approaches for abstraction and also with process discovery approaches.

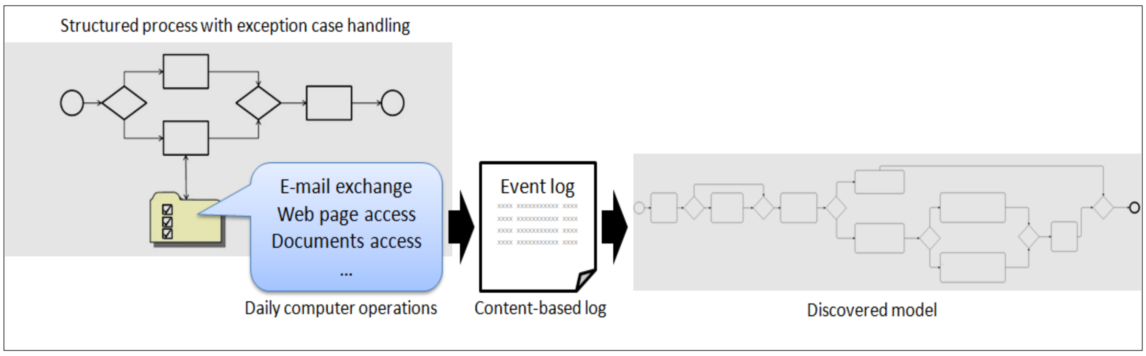


Figure 3.2 Discovery of process model from content-based log

In an application where processes cannot be pre-defined or by structured process with ad- hoc exceptions, the natural language descriptions have been shown as a common solution to associate with performed actions. Those daily computer operations, e.g. e-mail exchange, web pages and documents accesses, obviously do not produce well-structured logs. [40], [41], [42] and [43] attempt to map real-life content-based logs to structured logs which the overview on discovery of process model out of content-based log as illustrated in Figure 3.2.

The approach which identifies process instances and activity types for each event based on similarity of keywords and names contained in the unstructured log contents is presented in [40]. The clustering technique is used to discover and match events to similar process instances. Due to the fixed number of activity types in a particular workflow, a classification algorithm with Naive Bayesian classifier is chosen for discovery activity types for each event. The second clustering and classification are also applied on the initial event logs from the first identification round. When the initial results are good enough, the second identification also obtains better results. Otherwise, performance decreases. The result of conversion to structured logs can be used as an input to process mining later on. However, how to define a "sufficiently good" of a result is still an issue for future challenge.

In [41], the approach consists of context discovery, action discovery and process mining to constructs knowledge worker activity model from raw event logs. The context and action discovery are performed by k-means clustering algorithm. Then probabilistic deterministic finite automata has been performed to model the transitions between actions due to being more suited to low-level events in the data. However, noises may occur in the obtained model. Therefore, pruning the model is a practical solution to reduce complexity to final model while still maintaining useful coverage information.

[42] proposed an approach for identifying process trace from unstructured logs based on a supervised text classification and modifications of existing mining algorithm. Basically, a supervised text classification technique assigns a label to unseen data based on the labeled data in training data sets. In this case, the training data set was provided by domain experts who manually map unstructured log entries to process activities. The classifier consumes the training sets and generates process traces from unseen unstructured logs. These experiments show 90% accuracy in mapping unstructured logs to activities/traces. These process traces results can be employed in a process mining or a compliance checking later on. However, it is not sufficient to only identify activity names in process compliance, other relevant attributes (e.g. associated data flows) need also to be considered.

The explorative approach presented in [43] allows a process analyst to identify related process instances. The main idea is that of constructing process queries to filter related instances from database. Since input data in un-predefined processes is usually in free text fields, it should be with care so the process query is defined by user. The user is allowed to explore the database and choose only relevant processes. Those attributes for database exploration are currently available for manual filtering e.g. subject, description, status, stakeholder, year, source, task, first unit, last unit and participating unit. Once a process query is created, the selected data can be used in process mining phase to obtain a process model. The Heuristics Miner is chosen because it peforms well with noisy logs. The approach was applied on data from a Brazilian public organization and claimed that a reasonable discovered process model was obtained, therefore no further clustering was needed.

Table 3.2 Overview of transition from unstructured to structured process approaches

Author	Dealing with Spaghetti process	Dealing with daily content-based	Human involved	Approach
[38] Aalst et al. 2007	✓	-	-	Fuzzy Mining: consider high frequency of occurrences
[39] Bose et al. 2009	✓	-	-	Identify common behavior by loop patterns
[40] Geng et al. 2009	-	✓	-	Consider similarity of keywords also using clustering and classification algorithms.
[41] Štajner et al. 2010	-	✓	-	K-means clustering algorithm and pruning model to reduce complexity
[42] Desai et al. 2010	-	✓	✓	a supervised text classification technique and training data provided by domain experts
[43] Esposito et al. 2012	-	✓	✓	allows a process analyst to identify related process instances through queries

In conclusion, the complexity in spaghetti-like processes has been accounted in [38] and [39] but with different abstraction techniques. In [38], the behaviors with high significant or high frequency of occurrences will appear in the process model, whereas [39] focused on discovery of common pattern by identifying loop constructs patterns then replacing them with abstract entities. While the abstraction technique in [38] has been integrated with its mining process, the approach in [39] can be seamlessly integrated as a pre-processing step with other approaches for abstraction.

[40] identifies process instances and activity types for each event based on similarity of keywords and names contained in the unstructured log contents with clustering and classification algorithms. Similarly, [41] discovers context and action with clustering and combined with the application of probabilistic deterministic finite automata to model connections between actions. Moreover, [41] goes beyond to pruning the model in order to reduce complexity in the final model while still maintaining useful coverage information.

However, [42] and [43] have no need to perform clustering technique. Instead, the expert user has been partially incorporated. [42] identifies process trace from unstructured logs based on a supervised text classification. The training data set provided by domain experts who manually map unstructured subset of log entries to process activities. Also, [43] presents an explorative approach which allows a process analyst to be involved in identifying related process instances by constructing process queries to filter related instances. While [42] pointed out an outline of application of their approach, [43] has been integrated into the process mining tool called MANA which claims that this kind of exploration by a process analyst during mining would not be allowed in ProM framework. Because PROM aims to automatically construct a process models based only on an event log.

3.3 Scenario 3: Process Mining for Compliance Regulations in Unstructured Processes

Compliance checking is gaining importance as organizations need to show that operational processes are executed in a controlled manner while satisfying predefined regulations requirements [44]. Using process mining is a powerful method for compliance checking which detects deviations from organization's regulations.

***Compliance** refers to the knowledge of all regulations which are imposed on the organization and the adherence of these regulations. Furthermore, compliance encompasses the establishment of adequate processes as well as the control and documentation of conformance to the relevant regulation for internal and external stakeholders... Rath 2009* [74]

[45] proposes an incremental approach to check the conformance of a process model and an event log. At first, the fitness between the log and the model needs to be ensured. Then, the appropriateness of the model can be analyzed with respect to the log. Appropriateness can be evaluated from both a structural and a behavioral perspective. Several metrics have been defined for both fitness and appropriateness checking. So altogether they allow for the quantification of conformance. Although the modeled processes are based on the Petri nets, the results of this paper can be applied to any modeling language that can be equipped with executable semantics.

A new approach for relaxing the strict traditional compliance criterion is introduced in [46]. This approach attempts to address in the context of changing to a new process schema and process instances have to be migrated in adaptive process management system. However, with high strict compliance standards, not all process instances can be migrated. Therefore, new classes for relaxing strict compliances are introduced to increase the number of process instances migrated to a new process schema. The new relaxed classes mainly deal with loop constructs. They also present strategies for dealing with non-compliant instances such as rollback, delayed migration and adjust change operations.

In conclusion, [45] deals with compliance regulations by process mining techniques however their approach considered on well-structured which may not be suitable in very loosely structured process. Although, a new compliance classes introduced in [46] for adaptive systems, but they have not regarded the benefits of process mining.

3.4 Conclusion and Selection of Scenario

According to analysis of current adaptive case management and process mining approaches of three scenarios includes:

- Process mining for supporting knowledge workers in ACM
- Process mining for transition from unstructured to structured process
- Process mining for compliance regulations in unstructured process

There are various approaches which attempt to use process mining to support knowledge workers in ACM. Prediction and recommendation is a popular solution to support knowledge workers to make the best action for each situation. Time is the nearly first aspect in prediction approaches [11], [15] in order to estimate remaining time so that it helps knowledge worker to perform cases within a proper time duration and do not violate deadline or delay. Generating recommendations [12], [13], [14] is one of the most challenging approach which is based on past cases or running cases in time interval in order to provide a list of best next actions so that knowledge workers have guidelines in difficult situations.

Transforming from unstructured to structured process is another well-known challenge due to the high complexity of the event log produced from unstructured processes resulting in an incomprehensible process model and is hard to extract knowledge out from it. Many researchers intend to challenge this issue. There are primarily two kinds of approaches. The first [38], [39] is transforming spaghetti-like process to become more understandable. This approach considers mainly frequency and common patterns. While, the second approach [40], [41], [42], [43] also uses data mining technique and involves humans to discover a process model from content-based log generated by daily computer activities.

Compliance checking is very important in validating regulations in organizations. However, there are only a few approaches which are directly addressing compliance checking with process mining in unstructured processes. [45] and [46] both deal with compliance regulations but [45] is mainly for well-structured process and [46] is without the consideration of process mining.

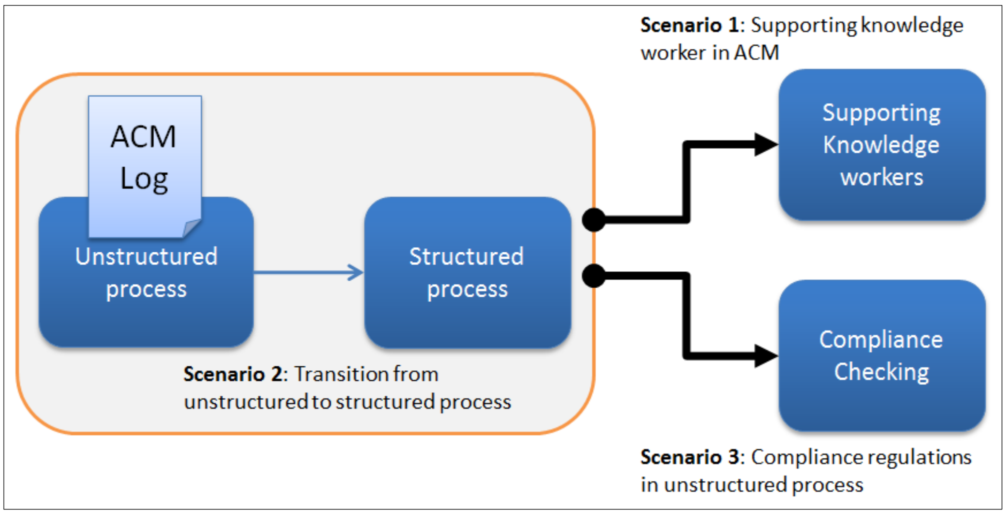

Scenario 1: Supporting knowledge worker in ACM
ACM Log
Unstructured process
Structured process
Supporting Knowledge workers
Compliance Checking
Scenario 2: Transition from unstructured to structured process
Scenario 3: Compliance regulations in unstructured process

Figure 3.3 The relationship between the three scenarios

Ultimately, the analysis of the three scenarios reveals that a connection between scenarios can be established. The scenario of transformation of unstructured to structured process can be the primary step for another two scenarios as shown in Figure 3.3. After obtaining an appropriate discovered process model, the model is reasonable in the form of a structured process. Consequently, the supporting system for knowledge worker can use the result model in the process of prediction and recommendation. Likewise, the scenario of compliance regulations in unstructured process can first apply the transformation of unstructured to structured process, then take the result of structured process as an input to perform in compliance checking step.

Regarding to truly importance of compliance checking in organizations and the fact that there are not many researches on the compliance checking in unstructured processes. Therefore, this master thesis purposes an approach which attempts to perform *compliance checking for unstructured processes using process mining*. The approach mainly consists of two parts. The first part uses process mining to transform unstructured to structured process, and the second part performs compliance checking based on the output process model from the first step. The overall approach for the prototype solution is shown in Figure 3.4.

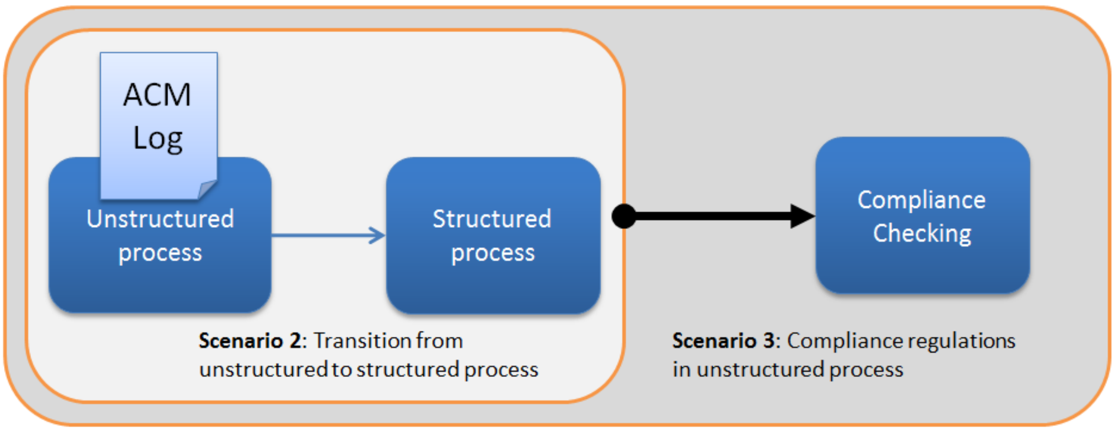

ACM
Log
Unstructured process
Structured process
Compliance Checking
Scenario 2: Transition from unstructured to structured process
Scenario 3: Compliance regulations in unstructured process

Figure 3.4 The selected scenario illustrated in the overall approach for the prototype solution

In the next chapter, the prototype approach as well as methods and technologies used in the approach will be described in more detail.

4 Architecture Design

In chapter 3, related work of adaptive case management and process mining combinations are analyzed. Particularly, the scenario is selected. This is the approach of compliance checking in unstructured processes using process mining.

In this chapter, the architecture of a prototype solution for the proposed approach is demonstrated. Besides, details stepwise of the architecture are explained.

4.1 Architecture Overview

In the master thesis, the approach to perform compliance checking in unstructured processes along with process mining is proposed. The approach combines two steps consisting of using process mining to transform an unstructured process to a structured process, and performing compliance checking on the output process model from the previous step.

The architecture overview illustrated in Figure 4.1 describes the approach of compliance checking in an unstructured process using process mining. Initially, process mining discoveries a structured process model from an unstructured event log. The result of the process model is imported to Oryx graphical editor. Rules are graphically defined. Then the defined rules are translated into a rule language. Afterwards, the defined rules validate with the original event log. All the violation are recorded and displayed visually on the process model.

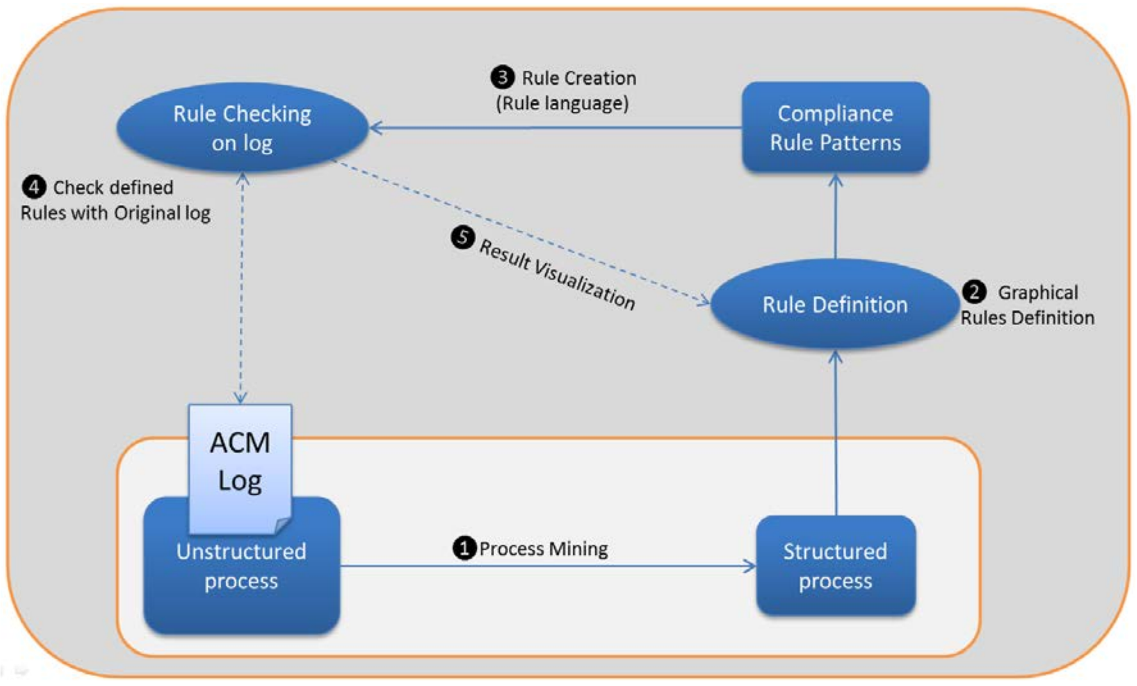


Figure 4.1 Architecure overview of compliance checking in unstructured process using process mining

4.2 Architecture Details

❶ Process Mining

An event log recorded in adaptive case management system, which is also considered as an unstructured process, is conveyed to the method of process mining. Process mining performs a process discovery technique on the event log to construct a structured process model.

❷ Rule Definition (Graphically defining rules on a process model)

The output of structured process model from the process mining step is imported to a process editor. Rules are graphically defined by assigning correct and incorrect rule patterns on the imported process model.

❸ Rule Creation (Rule language)

The compliance rule patterns assigned graphically in the previous step are converted into a rule language which will be used in the compliance checking process.

❹ Rule checking on log (Checking defined rules with original log)

The defined compliance rule patterns in the format of a rule language are validated against the entire original event log. All the violations are counted and process instances which violate the defined rules are recorded.

❺ Result Visualization

The violating behaviors recorded in the rule checking step are visualized as a highlight color on the process model. As well as in the information table which lists rules, violated process instances and total number of violated process instances.

5 Methods and Technologies

In chapter 4, the architecture overview is demonstrated. Besides, the proposed approach procedures are explained in five steps including process mining, rule definition, rule creation, rule checking on log and result visualization.

In this chapter, the methods and technologies which are used for realizing the approach of this work are analyzed. The methods and technologies are discussed in three aspects including process mining, rule definition and rule checking and compliance checking. At the end of each section, the suitable techniques for the approach are selected. The chapter winds up with a summary of the selected techniques shown in the architecture overview.

5.1 Process Mining

Process mining is used to discover hidden knowledge from event logs [2]. It is a well-known technique for understanding and improving business processes not only structured processes but also unstructured processes [7]. In this section, process mining techniques are examined to find out the best algorithm for a real-life log generated from adaptive case management system.

5.1.1 Analysis of Process Mining Techniques

There are a number of commercial process mining tools available on the market such as Celonis[4], Fluxicon[5] and myInvenio[6]. However, for academics purposes, PROM [47] is the most common process mining tool. PROM is developed by Eindhoven University of Technology. It is an open source framework with a variety of process mining techniques. PROM provides a platform which is easy to use for business users and also a framework for developers to implement new extensions or plug-ins. In this master thesis, PROM is selected for applying process mining techniques because PROM has a wide variety process mining techniques.

There are various process mining techniques available as PROM plug-ins. However, not all techniques are suitable for real-life data logs. In this analysis, heuristics miner and fuzzy miner are discussed because they are suitable for real-life logs. However, a simple technique like alpha algorithm is also discussed for demonstrating the basic idea of process mining algorithm.

- **Alpha algorithm**

  Alpha algorithm [16] is a basic technique for process discovery. It was one of the first algorithms capable of addressing concurrency. That is, alpha algorithm can be used to discover a process model which has loops, parallel paths and choices. The algorithm focuses on control flows or ordering relations of activities to capture behaviors. However, it ignores the frequency and other attributes such as resources and timestamps. The alpha algorithm is not robust for incomplete logs, so it is not applicable to real-life log. However, it is simple and commonly used as the fundamental concept for understanding process discovery algorithm. There are several extensions and modifications based on alpha algorithm such as alpha plus algorithm [48] and heuristics algorithm [19].

- **Heuristics Miner**

  Heuristics miner [19], [49] is a technique which is extended from alpha algorithm by considering frequency of patterns in logs. This means the heuristics algorithm can discover main behaviors and abstract from exceptional behaviors and noises (leaving out less important activities). Therefore it is suitable for real- life logs. During mining, case identifications and activities are considered as well as timestamps which are used to compute ordering of events. The

[4] http://www.celonis.de/
[5] http://www.fluxicon.com/
[6] https://www.my-invenio.com/

mining approach consists of three steps: constructing frequency table, inducting a dependency graph from the table, and generating workflow net from the table and the graph of the two previous steps. The output model is heuristics net which can be converted to other types of process models, such as Petri net for further analysis.

- **Fuzzy Miner**

  Fuzzy miner [20] aims to address with process complexity. It highlights significant information and hides less significant activities. It uses significance and correlation metrics to simplify less structured processes. The idea of simplification is that highly significant behavior is preserved, less significant but highly correlated behavior is aggregated into clusters, and less significant and less correlated behavior is abstracted. The significance and correlation can be configurable to adjust a process model to visualize in desire level. The output model is fuzzy model and it can animate behaviors of an event log. However, the fuzzy model cannot be converted to other types of process modeling.

5.1.2 Selection of Process Mining Techniques

In the previous chapter, the approach of compliance checking for unstructured processes using process mining is proposed. The first step of the approach is discovering from unstructured processes to structured processes. In this step, PROM is used for performing process mining to construct a process model from an unstructured process. Therefore, the process mining algorithm which is suitable for real-life logs or unstructured processes is needed.

Alpha algorithm is a basic technique which is able to deal with loops, parallel parts and choices however it is not robust for incomplete logs and not applicable to real-life log. Heuristics miner abstracts the process model based on frequency of activities which can extract main behaviors of the unstructured processes. Similarly, fuzzy miner aims to reduce complexity of the process by highlighting significant information and hiding less significant activities. Heuristics and fuzzy miners are suitable for unstructured processes. However, fuzzy miner constructs a fuzzy model which cannot be converted to other types of process modeling, while heuristics miner produces a heuristics net which can be converted to other types of process models for further analysis. In the work, BPMN is used for representing process models. Therefore heuristics miner is selected for the step of process discovery from an unstructured process to a structured process because it is suitable for a real-life log and it can be convert to other types of process models.

5.2 Rule Definition and Rule Checking

One of mandatory factors of business systems is business rules [50]. A process model tells what the right things to do, while business rules tell how to do the right things [51]. In this section, the meanings of rule in different aspects are discussed and the platform for implementation graphical rule definition and compliance checking are examined and selected.

5.2.1 Meaning of Business Rule

Business rule gives prescribed guide for actions or business behaviors. Business rules are often seen in contracts, regulations and warranties. In business process, conditions or constrains for example, activities ordering, limits and alerts can be expressed through business rules. The important thing is that those rules are needed to be interpreted and applied to the system of an organization. In other words, business rules are not technical specifications rather they are interpreted in simple statements clarifying the meanings from business people so that developers can use it for implementation [51].

The general definition of a business rule is given by Business Rules Group as follows:

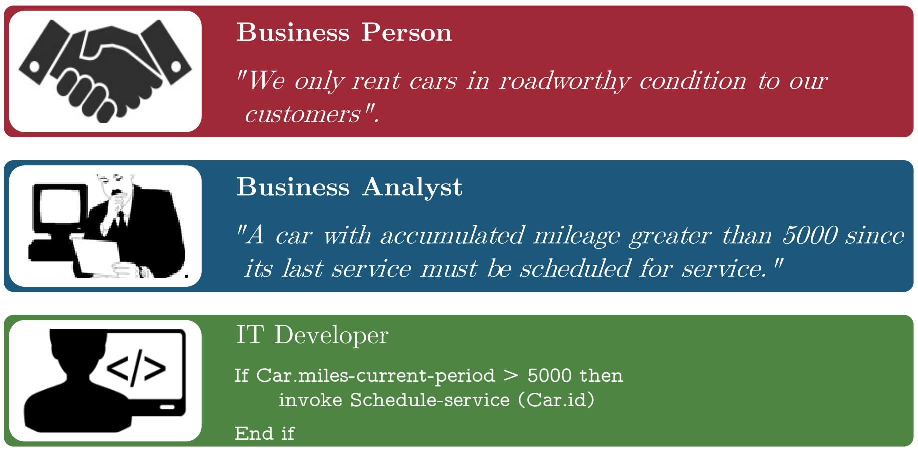
*A **business rule** is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business...Business Rules Group 2000* [52]

Business Rules Group classifies a business rule into two perspectives including business perspective and information system perspective. The clarifications of the both perspectives explained as follows:

*From the **business perspective**: "... a business rule is guidance that there is an obligation concerning conduct, action, practice or procedure within a particular activity or sphere. Two important characteristics of a business rule: (1) there ought to be an explicit motivation for it, and (2) it should have an enforcement regime stating what the consequences would be if the rule were broken"...Business Rules Group 2000* [52]

From the ***information system perspective:*** *"… a business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure, or to control or influence the behavior of the business"…Business Rules Group 2000* [52]

In Figure 5.1 demonstrates an example of a rule formulating in different roles including business person, business analyst and IT developer. A business person gives a rule general statement of direction for an organization. The general rule statement is interpreted by a business analyst to a business rule statement which is a declarative statement with some constrains of the business. An IT developer implements an expression of the business rule in a specific computer language in the organization's system.



Business Person

"We only rent cars in roadworthy condition to our customers".

Business Analyst

"A car with accumulated mileage greater than 5000 since its last service must be scheduled for service."

IT Developer

```
If Car.miles-current-period > 5000 then
      invoke Schedule-service (Car.id)
End if
```

Figure 5.1 Rule formulating in different roles (adapted from [52])

5.2.2 Platform Analysis for Rule Definition and Rule Checking

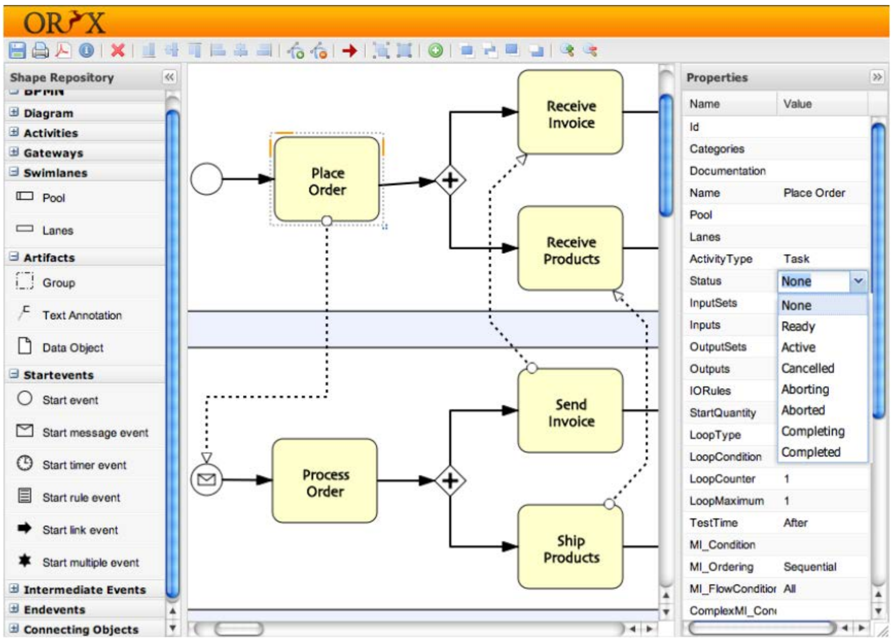
Business rules are determined commonly among business people and they are often in plain text business document. Business analysts extract those business rules into IT specifications. The technicians refer to the IT specifications and implement rules into the system. However, some information may be missing or misunderstood on the way during translating from business documents into IT specifications. Therefore, one alternative to minimize the confusions is a graphical process modeling tool, which business people and business analysts can understand. Besides, they are able to define business rules directly to the system by themselves.

5.2.2.1 Oryx process editor

A huge number of graphical process modeling tools either web-based or stand-alone applications are available such as Activiti[7], Camunda[8] and Signavio[9]. Oryx [53] is one of the most common academic open source platform for graphical process modeling. Oryx is initially started by Business Process Technology research group at the Hasso Plattner Institute of IT Systems Engineering at the University of Potsdam. Oryx is a web-based process modeling editor which supports in various modeling languages for example BPMN, EPC and Petri nets. The user interface of BPMN modeling in Oryx can be seen in Figure 5.2.

The architecture of Oryx illustrates in Figure 5.3. Oryx is a web-based process editor. It runs through a web browser without an additional software installation. Oryx consists of Oryx core which are stencil sets and plugins used for creating process models on web browser. The so-called stencil sets are set of graphical icons which are used for building process models. The plugins may process only on client side or call functions in Oryx backend. New features can be implemented via plugins. Displaying multiple languages is supported by using I18N which can be adapted to specific local languages. The created process models are saved in repository and can be loaded for future usages. With data portability, Oryx has the ability to import and export process models from and to different platforms.

[7] http://activiti.org/
[8] https://camunda.com/
[9] http://www.signavio.com/de/



Name	Value
Id	
Categories	
Documentation	
Name	Place Order
Pool	
Lanes	
ActivityType	Task
Status	None
InputSets	
Inputs	
OutputSets	
Outputs	
IORules	
StartQuantity	
LoopType	
LoopCondition	
LoopCounter	1
LoopMaximum	1
TestTime	After
MI_Condition	
MI_Ordering	Sequential
MI_FlowConditior	All
ComplexMI_Con	

Figure 5.2 Oryx interface for BPMN modeling. Drag and drop elements are on the left side and the properties for each element are on the right side [53].

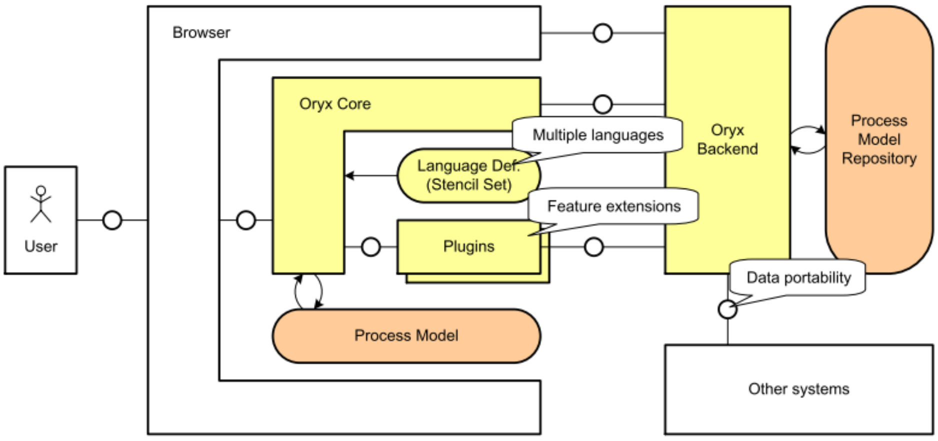


Figure 5.3 Oryx architecture [53]

Oryx is extensible via a plugin mechanism. Almost all features are implemented as plugins. The functions are written in JavaScript for client side and in Java for server side. To implement a new stencil set, the basic three technologies are applied:

- JavaScript Object Notation (JSON) is used for descripting a stencil, stencil's properties and rules.

- Scalable Vector Graphics (SVG) is used for representing graphical objects
- JavaScript is used for implementing functions for stencil set actions on client side

5.2.2.2 Fraunhofer IAO Process Editor

Fraunhofer IAO process editor is another tool which based on Oryx. The interface of the Oryx extended by Fraunhofer IAO is shown in Figure 5.4.

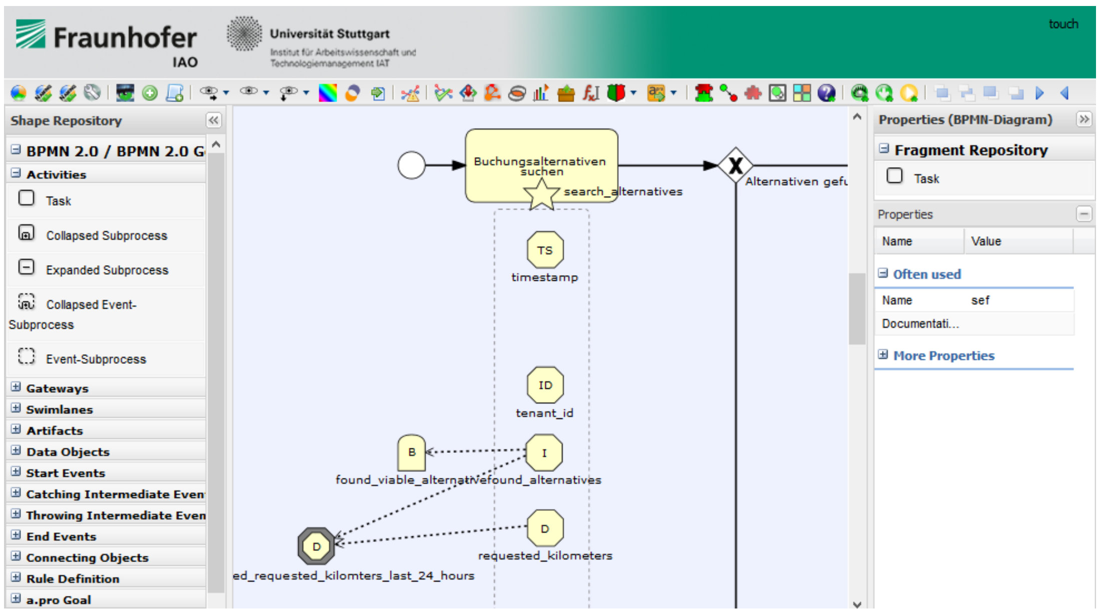


Figure 5.4 Interface of the Oryx extended by Fraunhofer IAO

The Oryx modified by Fraunhofer IAO has been implemented a number of new stencil sets and plugin features. It has been also used in a various number scientific researches by Fraunhofer IAO and University of Stuttgart. For instance, Business Process Optimization In Cross-Company Service Networks [54]. Integrating Compliance Requirement across Business and IT [55] and A Model-Driven Approach for Business Process Monitoring [56].

Everyone is invited to contribute new process modeling languages and features to Oryx. Several number of business process modeling tools either for academics or commercial purposes extended from Oryx such as Signavio. Signavio opens for academic uses and it has also a commercial version providing all features needed for professional business purposes.

5.2.3 Selection of Rule Definition and Rule Checking Platform

The aim of this work is compliance checking for unstructured processes. To validate compliances, rules are needed. Rules should be defined in understandable manners. One alternative to achieve more comprehension is to define rules graphically. According to graphical process modeling tools analysis, the web-based open source platform like Oryx, provides the possibility to extend the tool with new extensions or plugin and stencils sets in order to enhance the existing functionalities. With a plugin, it is considered as a separate module so that it can be added or removed without any effects to other parts of the tool.

Furthermore, this master thesis is a cooperation project between Fraunhofer IAO and IAAS institute of University of Stuttgart. Besides, the graphical process modeling tool by Fraunhofer IAO is developed based on Oryx platform. This powerful tool has been used in a number of remarkable researches and developments relating in business process management area. With the modified version by Fraunhofer IAO and the extensibility of Oryx, the implementation of the prototype solution for this project inherits direct benefits. The implementation can utilize the Fraunhofer IAO process modeling tool based on Oryx as the main platform for the prototype solution and develop new plugins for the graphical rule definition and rule checking for compliance. Therefore the Oryx modified by Fraunhofer IAO is best platform for this work.

5.3 Compliance Checking

Compliance checking is a process of investigating whether defined conditions or constraints are met. Without compliance checking, violating behaviors against an organization's regulation may be disregarded. Therefore, the core challenge is to compare the defined behaviors (set of rules) to the actual behaviors (event log). This challenge is realized in this work thus the approach of compliance checking by checking rules on an event log in unstructured processes is introduced. In this section, compliance checking concept and techniques are discussed and the suitable technique is selected at the end of the section.

5.3.1 Analysis of Compliance Checking Techniques

The increasing number of compliance requirements in organizations makes it complicate to manage, therefore a software system for compliance management is necessary [57]. A complete compliance management process is suggested by [55] as shown in Figure 5.5.

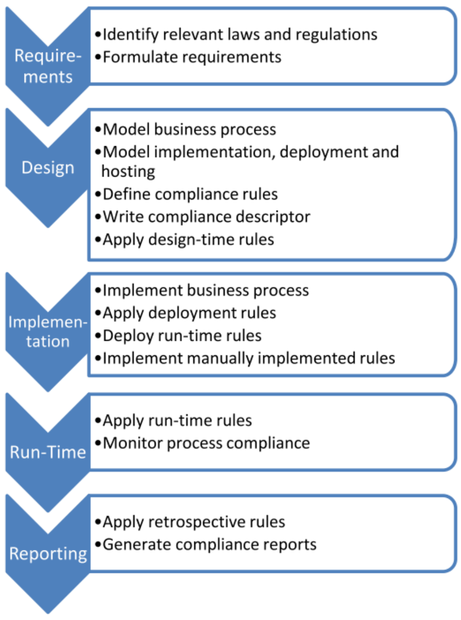

Require-ments
•Identify relevant laws and regulations
•Formulate requirements
Design
•Model business process
•Model implementation, deployment and hosting
•Define compliance rules
•Write compliance descriptor
•Apply design-time rules
Implemen-tation
•Implement business process
•Apply deployment rules
•Deploy run-time rules
•Implement manually implemented rules
Run-Time
•Apply run-time rules
•Monitor process compliance
Reporting
•Apply retrospective rules
•Generate compliance reports

Figure 5.5 Compliance Management Process [55]

The compliance management process suggested by [55] integrates with the generic compliance descriptor. First, compliance *requirements* such as relevant laws and regulations are identified which are often formulated in plain speech.

At ***design*** phase, a business process model is constructed and implementation, deployment and hosting are planned. The compliance rules are written in a compliance descriptor according to the compliance requirements from the previous step. Also the design-time compliance descriptor rules are applied during this phase.

The ***implementation*** can proceed, if all the design-time rules are valid. The deployment rules are applied. The runtime rules are deployed for using during runtime. Some compliance rules in compliance descriptor may need to be implemented manually in the case that they are not compatible with the existing technology.

During ***runtime***, the business process is executed, the runtime rules can be applied and process compliance can be monitored. After execution, the retrospective rules are applied and compliance reports can be generated. At this stage, the compliance descriptor consisting of compliance rules have been applied. The final report covering all steps can be generated, however, intermediate reports before process execution may be necessary.

From compliance management process, the types of compliance checking can be classified into two types, Forward compliance checking and Backward compliance checking as illustrating in Figure 5.6.

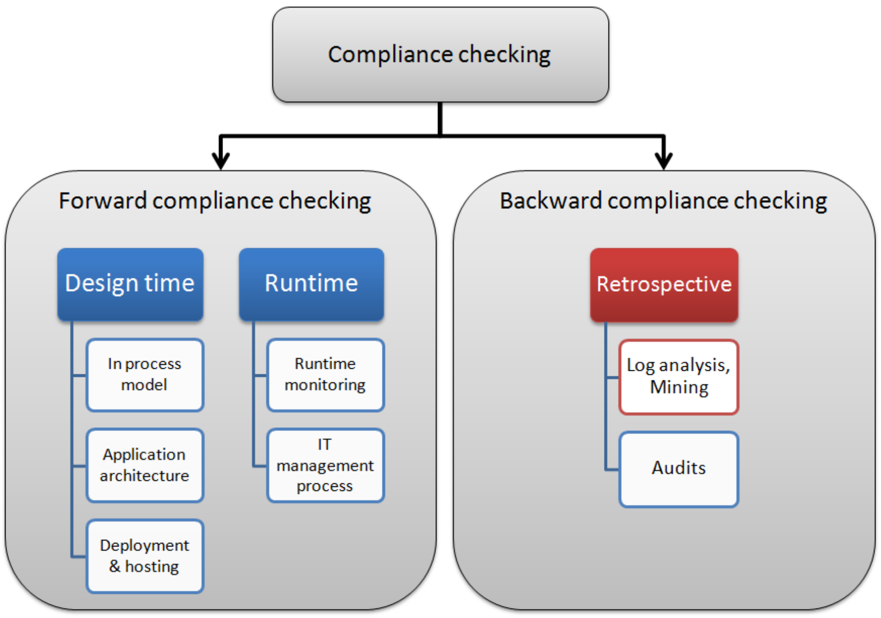


Figure 5.6 Types of compliance checking (adapted from [58] , [59])

5.3.1.1 Forward Compliance Checking

Forward compliance checking aims to prevent non-compliant behaviors by verifying rules before execution (design time) or during execution (runtime).

- At ***design time***, the compliance checking focuses on the compliance in process models, verifying application architecture and ensuring deployment and hosting [59]. Some guidelines or model checking between a process model and given compliance constraints during process modeling phase may be used to eliminate non-compliance behaviors before deploying the process model [58]. Several approaches attempt to get rid of non-compliant causes at design time. For example, [60] introduces the compliance patterns approach which compute the deviation of a process model to given compliance patterns. [61] proposes a formal framework which consists of definitions and a set of properties. A process implementing the framework has to must meet the given properties.

- During ***runtime***, the compliance checking targets to verify regulations which are not available at design time but present at runtime [62]. It can be further classified into enforceable and non-enforceable rules. The enforceable rules can be prevented before violations occurring by monitoring, measuring or comparing to applicable service level agreements. The non-enforceable rules can be monitored and reports the violating behaviors but cannot prevent them [63]. Several approaches demonstrate compliance checking during runtime. For example, [56] proposes a model-driven approach for generating a monitoring infrastructure. The approach allows users to define a monitoring model and transform it into event rules. During runtime, the event rules are evaluated using a Complex Event processing (CEP) mechanism. [64] introduces an on-the-fly auditing based on Petri nets. In running executions, if every transition can be fired, the execution is assumed to be compliant.

5.3.1.2 Backward Compliance Checking

Backward compliance checking or retrospective analysis targets to detect whether there is a violation with certain rules by analyzing the logs of business processes executions. The logs can be analyzed, applied with process mining techniques or using in-auditing to find out non-compliant behaviors [59]. The backward compliance checking is mainly used for controlling and reengineering purposes [62]. Several techniques for compliance checking have been introduced to detect deviations between event logs and process models. For example, [45] develops conformance checking plug-in in PROM. The approach measures the degree of an event log matching to the process model. [65] implements a LTL checker plug-in also in PROM. The idea is, defining business rules using Linear Temporal Logic (LTL) and verifying them against process execution logs.

5.3.2 Selection of Compliance Checking Techniques

According to the selected scenario which is compliance checking for unstructured processes using process mining. An event log which is produced from execution of adaptive case management system is considered. Therefore, the approach focuses on the history of process executions. This means, the approach is be within the scope of *backward compliance checking or retrospective analysis* (marked in red on Figure 5.6) due to targeting to detect deviations in event logs left by process executions. In compliance checking by checking rules on a log, certain rules are defined and validated with the original event log to discovery non-compliant behaviors.

In [56], Complex Event processing (CEP) mechanism is used to monitor generated rules in real-time. Interestingly, CEP engine can be used in offline setting for backward compliance checking as well. In this work, *Esper Complex Event processing* is selected due to the abilities in analyzing high volume of event series and for allowing customized triggers for detecting violations among events (See section 6.5). In the case of compliance checking for this work, the event series are from an event log of ACM system and the customized triggers are sets of graphical defined rules assigning on a process model. Then, the graphical defined rules are interpreted into Esper rule language and used for validating an event log to detect violations.

Further, the approach of compliance checking for unstructured processes is integrate with the concept of compliance descriptors [55] (See Figure 5.5) which is further described in [66].

5.4 Conclusion of Methods and Technologies

Methods and Technologies for compliance checking in unstructured process using process mining are illustrated in Figure 5.7.

In the first step of the approach, *PROM* is a selected tool for the discovery a structured process from an unstructured process with heuristics miner. The result process model is then imported into Oryx process editor for compliance checking. Due to different BPMN versions between PROM and Oryx, the PROM to Oryx converter is need to be implemented to import the discovered process model from PROM into Oryx (See section 6.3.1).

In *Oryx process editor*, compliance checking functionalities are developed with the concept of backward compliance checking. A new stencil set is developed for the graphical rule definitions. Then, the graphical rules will be translated into a rule language – in this work Esper rule is selected. The Esper rule is expressed in Event Process Language (EPL). Lastly, the original log is loaded to validate with defined Esper rules by Esper Complex Event Processing (CEP) mechanism. The result visualization is implemented to display violating behaviors on the imported process model and also report violation details including total number of violations and violating process instances.

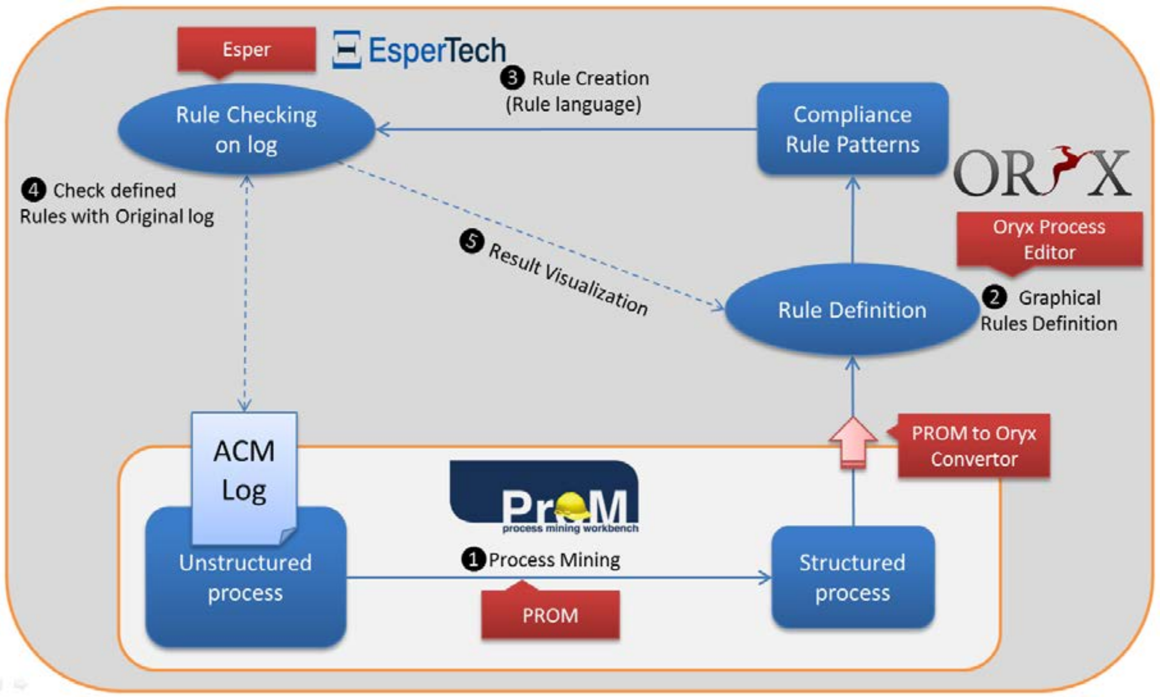


Figure 5.7 Overview of Methods and Technologies employing in the compliance checking in unstructured process using process mining

In the next chapter, the implementation of the prototype solution is demonstrated to proof the approach of compliance checking in unstructured process using process mining.

6 Implementation

In chapter 5, methods and technologies using in the approach of compliance checking in unstructured processes using process mining are selected and explained.

In this chapter, implementation of the prototype solution is demonstrated as well as pointing out the challenges encountered during implementation.

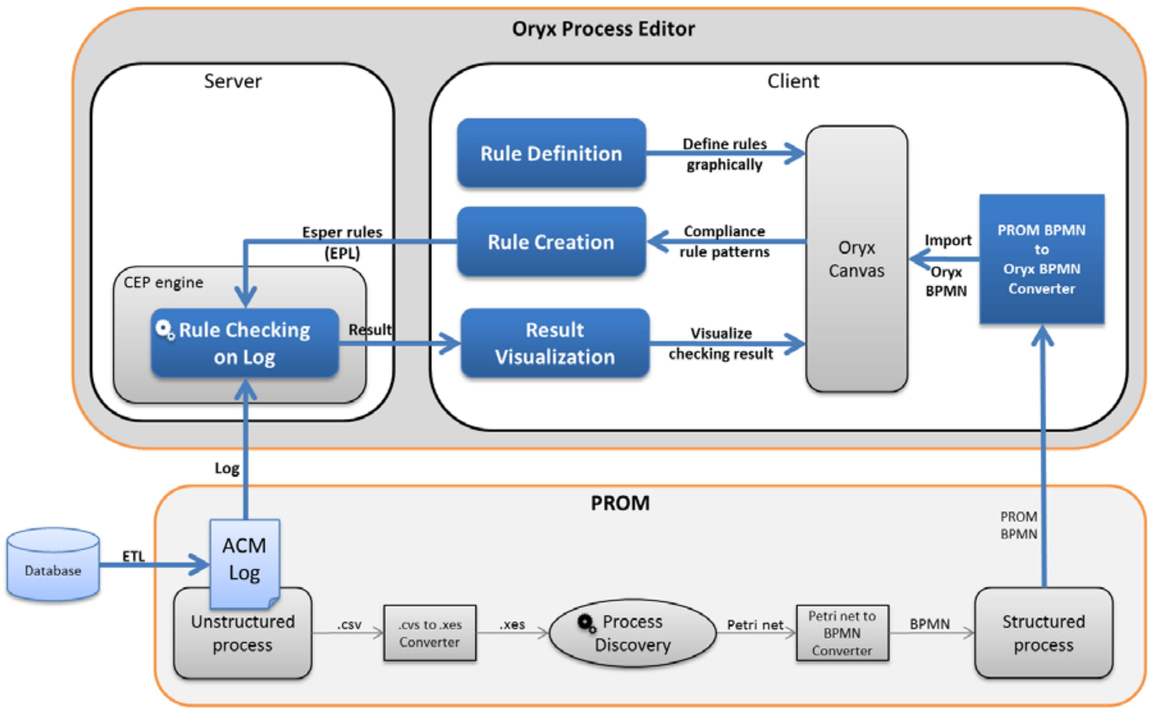


Figure 6.1 The detail architecture detail of compliance checking in unstructured process using process mining

As a proof of the approach, PROM as a process mining tool and Oryx process editor modified by Fraunhofer IAO are employed. The process mining technique in PROM is used to discovery a structured process from an unstructured process. Also, new features for compliance checking are implemented in Oryx process editor. The detail architecture of the prototype solution of compliance checking in unstructured process using process mining illustrates in Figure 6.1.

The prototype consists of six parts including Event Log Preparation, Performing Process mining, Oryx Rule Definition, Rule Creation, Rule Checking on Log and Result Visualization as shown in Figure 6.1. The components represented in dark blue squares are newly developed in this master thesis. The components represented in light blue are the data input. The prototype utilizes the components represented in grey are functionalities which are already available in Oryx and PROM.

The historic data and knowledge worker's actions from adaptive case management system are recorded in the database. The event log in the form of CSV file is extracted from the database by the classic ETL process.

The event log is fetched to PROM. Before performing process mining, the event log is converted into XES format by PROM's plugin. The process discovery constructs a structured process from the unstructured process (event log). The structured process in form of Petri net is transformed into BPMN.

The output process model from PROM is in form of BPMN. Due to BPMN version conflict the PROM BPMN is mapped into Oryx BPMN version before importing to Oryx platform. Then, rules are defined graphically on the imported process model displaying in Oryx canvas. Rule creation translates the graphical rules into Esper rule language expressed in Event Process Language (EPL). The Esper rules and original event log are fetched to Complex Event Processing (CEP) engine to perform rule checking on log. CEP engine notifies an event trigger whenever there is a violating behavior with the defined rules. All the violations are recorded then they are visualized the violated paths on the process model and detail violations in an information table.

6.1 Event Log Preparation

The initiative input of the approach is an event log. In this context, the event log is generated from adaptive case management system. The process execution history and every knowledge workers' actions are stored in the database. In this work, the classic ETL process (extraction, transformation and loading) is employed for event log preparation. An example tool is Pentaho Data Integration (PDI, also shown as Kettle) which is a leading open source ETL tool on the market. It allows data manipulations across multiple sources. The typical functions are such as reading, refining, transforming, and writing data to various formats [67].

First, logs of knowledge worker's actions recorded in the database are extracted. Then, the transformation function manipulates data from database and cleans the data which aims to filter only the proper data to the target. Also, the data is transformed into the destination format. In the work, Comma Separated Values (CSV) is used. Now, the event log recorded in database of ACM system is presented as a to CSV file and it is ready to be loaded into process mining step.

6.2 Performing Process Mining

After preparing the event log, now it is ready to use in process mining step. In this approach, PROM is selected as the process mining tool due to a wide variety supporting of process mining techniques. In this step, PROM performs heuristics miner to discover a structured process form an unstructured process.

XES is a format of event logs which PROM commonly uses. In log preparation, the event log is prepared in CSV format. However PROM has a converter plugin ready to use for converting CSV log into XES format.

Some outstanding process discovery techniques are experimented including Fuzzy miner and Heuristic miner. They are notable algorithms used in practical applications and real-life log. The same prepared event log is fetched into each process discovery technique for comparing among techniques.

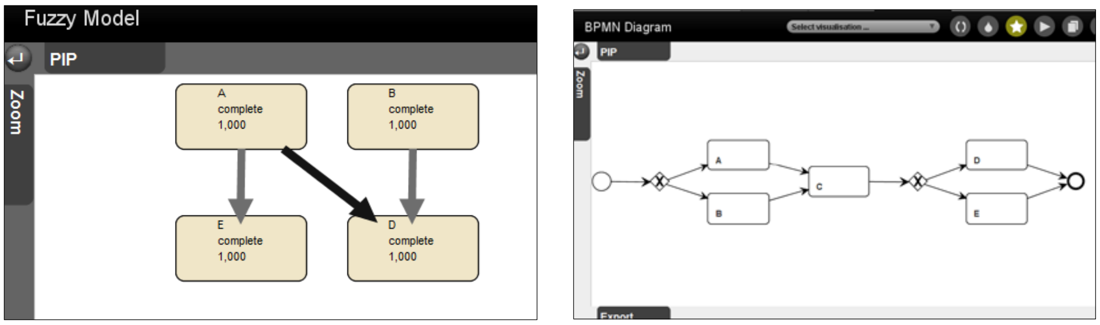


Figure 6.2 Results of process discovery produced by Fuzzy miner (left) and Heuristic miner (right)

In Figure 6.2, Fuzzy miner produces the output model which can be adjusted at a desired level of abstraction. However, the fuzzy model cannot be converted into other types of process modeling languages. Heuristics miner focuses on the frequency of patterns so the main pattern behaviors in unstructured processes can be extracted. Also, the output of heuristics miner is possible to obtain BPMN out of the discovered process models. Therefore, heuristics miner is the process mining algorithm which is selected for the work.

6.3 Rules Definition

After performing process mining, a process model in form of BPMN is obtained. The output discovered process model has to be imported into Oryx process editor. However, there is a conflict between BPMN version of PROM and Oryx. Therefore, a converter for transforming BPMN PROM into BPMN Oryx is needed to be developed so that the output discovered process model is able to be import into Oryx process editor

6.3.1 PROM to Oryx converter

BPMN PROM into BPMN Oryx converter is developed to convert BPMN generated by PROM into a format suitable for import in the Oryx platform. The conflict is because of the difference of definitions which make the differences on structures of PROM BPMN and Oryx BPMN. The definition tags of both BPMN are shown in Table 6.1.

Table 6.1 The comparison of definitions between BPMN of PROM and Oryx

PROM

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
targetNamespace="http://www.omg.org/bpmn20"
exporter="ProM. http://www.promtools.org/prom6"
exporterVersion="6.3"
xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL
BPMN20.xsd">
...
```

Oryx

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schema.omg.org/spec/BPMN/2.0"
xmlns:bpmndi="http://bpmndi.org"
typeLanguage="http://www.w3.org/2001/XMLSchema"
expressionLanguage="http://www.w3.org/1999/XPath"
targetNamespace="http://www.omg.org/bpmn20"
id="oryx_6e741aa1-8396-40f5-8701-52e6582e82ed">
...
```

Figure 6.3 illustrates the difference of BPMN structures generated by PROM and Oryx. It also shows the mapping fields for converting from PROM into Oryx BPMN structure.

PROM BPMN Structure has mainly two parts in its `<definitions>` which consists of `<process>` and `<bpmnd:BPMNDiagram>`. In Oryx BPMN structure, it has also two parts but it contains different elements. It consists of `<process>` and `<bpmnd:processDiagram>`. Inside `<process>` has `<laneSet>` and bpmn elements. Beside, `<bpmndi:processDiagram>` has `<bpmndi:laneCompartment>` and `<bpmndi:sequenceFlowConnector>` elements. These cause the problem in importing function of Oryx due to mismatching in BPMN structures.

The converter is developed to solve this issue by extracting contents in each field from PROM BPMN and organizing in the new structure which matches with Oryx BPMN structure.

The `<process>` of PROM can be mapped to `<laneSet>` and bpmn elements in Oryx's `<process>`. The `<lane>` inside `<laneSet>` has the list of `<flowElementRef>` which contains all bpmn element ids of a diagram. All the element ids are extracted by the details of each element in PROM BPMN. These element ids are listed in `<flowElementRef>` of Oryx BPMN. For bpmn elements themselves, they can be directly mapped all elements in PROM's `<process>` into the second part of Oryx's `<process>.`

The <bpmndi:BPMNPlane> which is a element of <bpmnd:BPMNDiagram> in PROM, is mapped to <bpmndi:laneCompartment> and with its shape type e.g. <bpmndi:eventShape>, <bpmndi:gatewayShape> or <bpmndi:activityShape>. For <bpmndi:BPMNEdge> element, they can be directly mapped into <bpmndi:sequeceFlowConnect>.

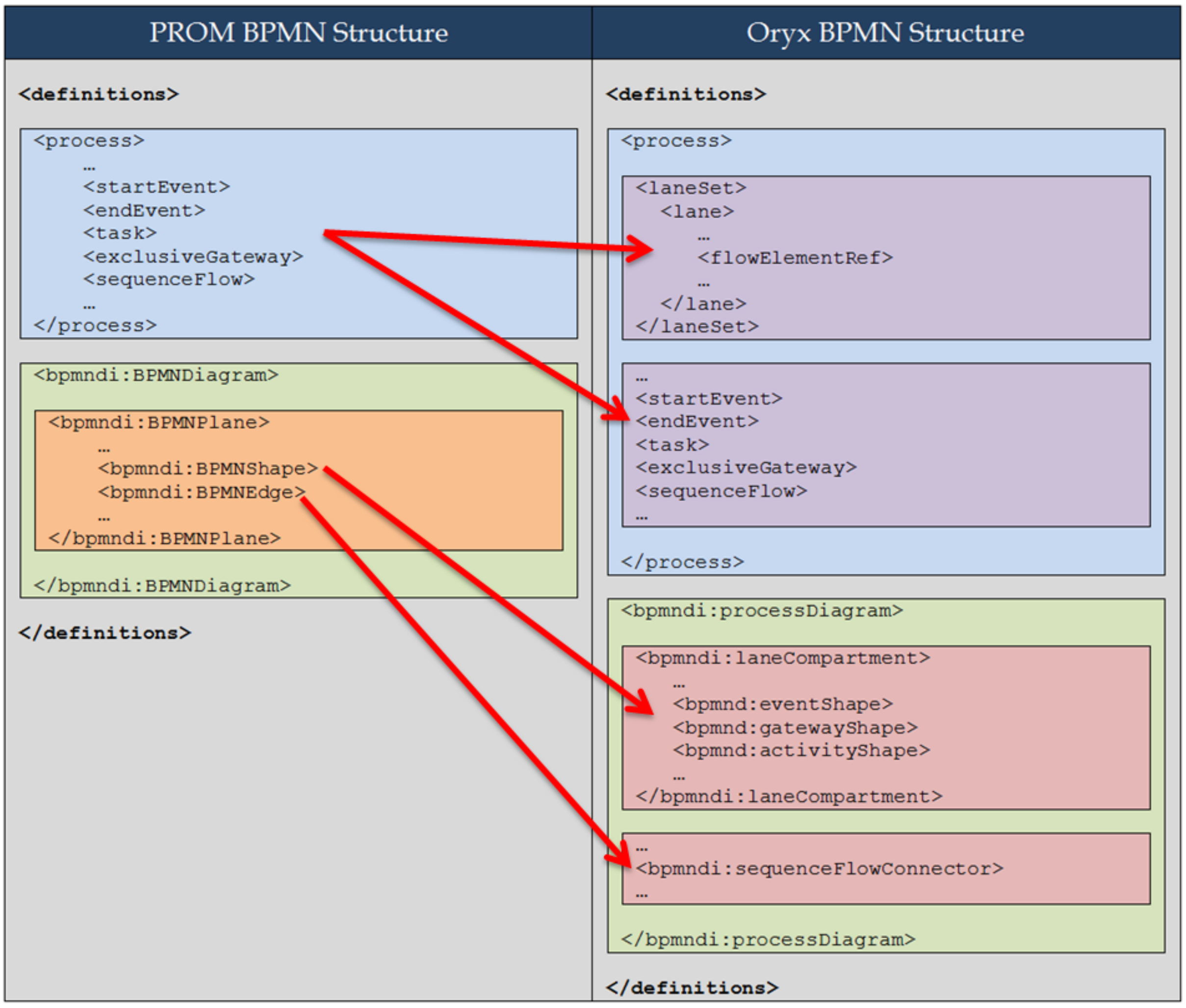
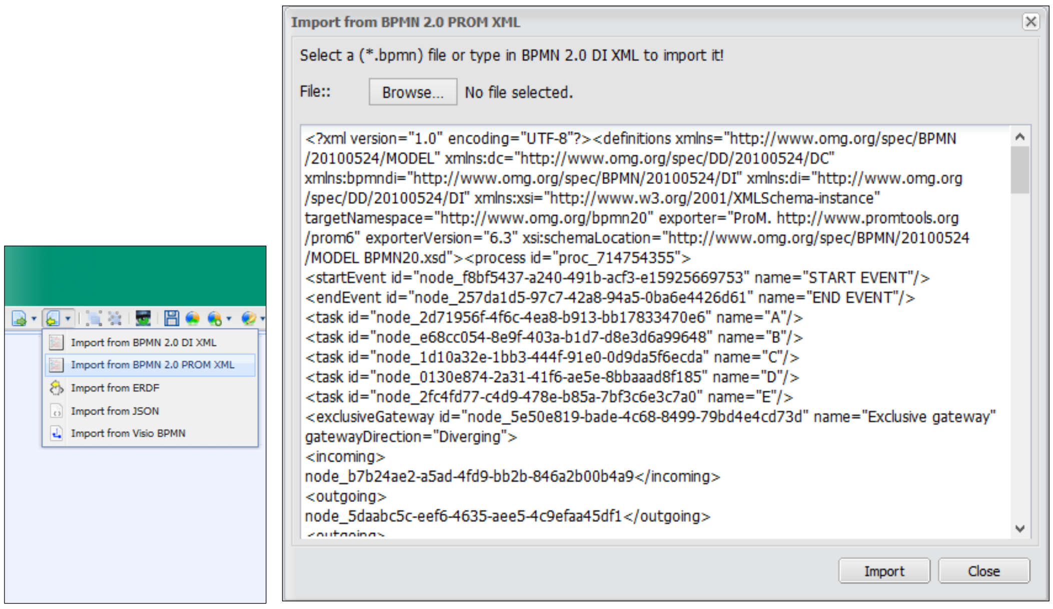


Figure 6.3 Comparison of PROM and Oryx BPMN structure and mapping fields between the two structures

The converter for transforming PROM BPMN into Oryx BPMN is developed. So now, the discovered process model from PROM can be imported into Oryx platform. Figure 6.4 shows how a process model generated by PROM can be imported to Oryx and Figure 6.5 shows that the discovered process model can be perfectly displaying in Oryx process editor.


Import from BPMN 2.0 DI XML
Import from BPMN 2.0 PROM XML
Import from ERDF
Import from JSON
Import from Visio BPMN
Import from BPMN 2.0 PROM XML
Browse...
Import
Close

Select a (*.bpmn) file or type in BPMN 2.0 DI XML to import it!

File:: No file selected.

```
<?xml version="1.0" encoding="UTF-8"?><definitions xmlns="http://www.omg.org/spec/BPMN
/20100524/MODEL" xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI" xmlns:di="http://www.omg.org
/spec/DD/20100524/DI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
targetNamespace="http://www.omg.org/bpmn20" exporter="ProM. http://www.promtools.org
/prom6" exporterVersion="6.3" xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524
/MODEL BPMN20.xsd"><process id="proc_714754355">
<startEvent id="node_f8bf5437-a240-491b-acf3-e15925669753" name="START EVENT"/>
<endEvent id="node_257da1d5-97c7-42a8-94a5-0ba6e4426d61" name="END EVENT"/>
<task id="node_2d71956f-4f6c-4ea8-b913-bb17833470e6" name="A"/>
<task id="node_e68cc054-8e9f-403a-b1d7-d8e3d6a99648" name="B"/>
<task id="node_1d10a32e-1bb3-444f-91e0-0d9da5f6ecda" name="C"/>
<task id="node_0130e874-2a31-41f6-ae5e-8bbaaad8f185" name="D"/>
<task id="node_2fc4fd77-c4d9-478e-b85a-7bf3c6e3c7a0" name="E"/>
<exclusiveGateway id="node_5e50e819-bade-4c68-8499-79bd4e4cd73d" name="Exclusive gateway"
gatewayDirection="Diverging">
<incoming>
node_b7b24ae2-a5ad-4fd9-bb2b-846a2b00b4a9</incoming>
<outgoing>
node_5daabc5c-eef6-4635-aee5-4c9efaa45df1</outgoing>
```

Figure 6.4 Importing BPMN generated by PROM to Oryx process editor

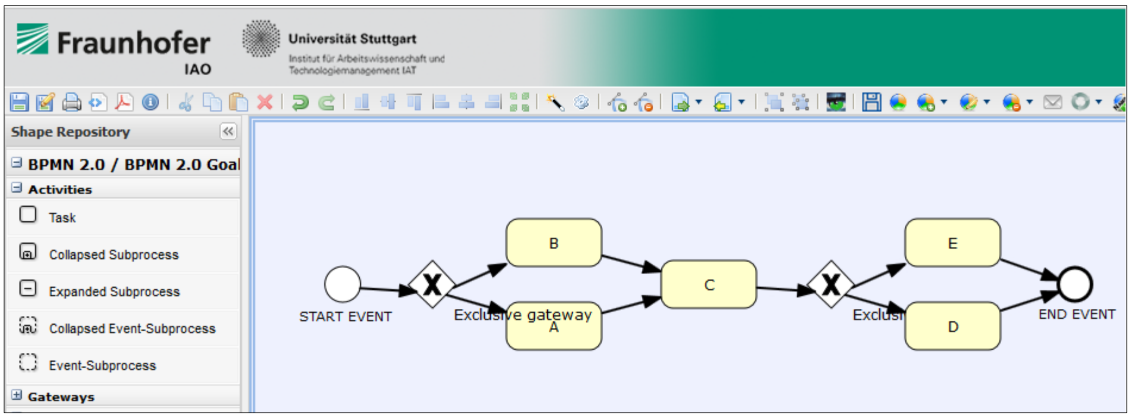


Figure 6.5 PROM discovered process model displayed in Oryx process editor

6.3.2 Graphical Rule Definition

The rule definition extension consists of three new icons which are *Correct pattern*, *Incorrect pattern,* and *Indirect flow* (See Figure 6.6). The *Correct* and *Incorrect pattern* icons can be attached onto sequence flows of a process model to define rules whether the behaviors must be done (correct pattern) or must not be done (incorrect pattern).

Indirect flow is created to define an additional sequence flow which does not exist in an imported process model. Indirect flow gives the meaning of indirect following. If an indirect flow connects activity A to B, then B has to follow A but B does not need to immediately follow after A. For example, A can be followed by C and then B, and it still complies with the rule.

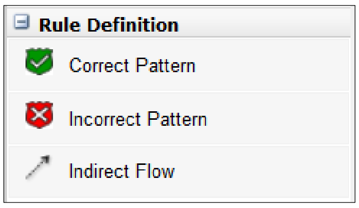
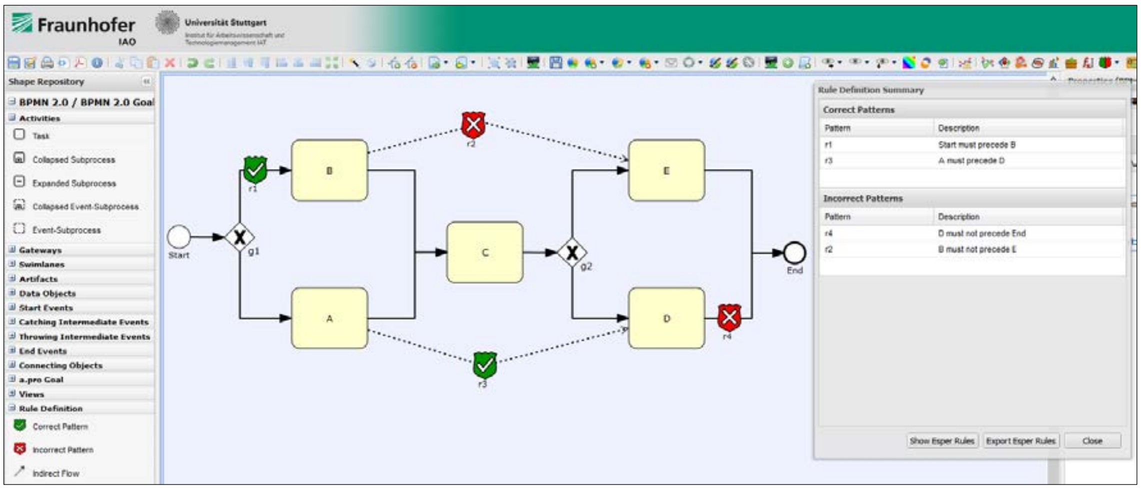


Figure 6.6 Rule definition icons consists of Correct pattern, Incorrect pattern, and Indirect flow



Rule Definition Summary

Correct Patterns

Pattern	Description
r1	Start must precede B
r3	A must precede D

Incorrect Patterns

Pattern	Description
r4	D must not precede End
r2	B must not precede E

Figure 6.7 The graphical rule definition in Oryx process editor and the rules summary panel

Figure 6.7 illustrates an example of rule definition in Oryx.

- Rule *r1* is attached on sequence flow from *Start* to *B* with a correct pattern icon. This means *B* must immediately follow after *Start*.
- Rule *r2* is attached on the indirect flow from *B* to *E* with an incorrect pattern icon. This means *E* must not follow after *B* either immediately or later.

- Rule *r3* is attached on the indirect flow from *A* to *D* with a correct pattern icon. This means *A* must precede *D* but *D* does not need to immediately follow after *A*.
- Rule *r4* is attached on the control flow from *D* to *End* with an incorrect pattern icon. This means *End* must not immediately follow after *D*.

6.4 Rule Creation

After rules have been graphically defined, these graphical rules will be translated into a rule language. In this work, Esper rule is selected because Esper is an open-source Java-based framework, and it is commonly used for Complex Event Processing (CEP) to analyze event series for detecting situations among events which can be used for compliance checking. Esper is expressed in Event Process Language (EPL) which is a SQL-like language with for example SELECT, FROM, and WHERE clauses [68].

The rule creation of graphical rules to esper rules generates rule statements which state the violating behaviors of the defined rules. Table 6.2 demonstrates all four possibilities of rule definitions and their rule creations into the rule language. For instance, defining a graphical rule *r1* as "*B* must immediately follow after *Start*", the rule statement will be described the defined rule (if *Start*, then *B*) into the form of negation pattern (if *Start*, then not *B*) which means *B* must not follow after *Start*.

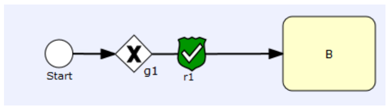
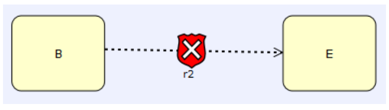
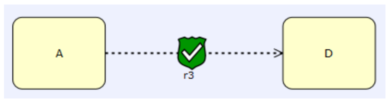
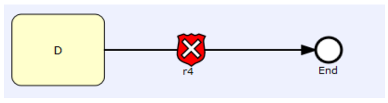
Similarly, if the graphical rule *r4* is defined as "*End* must not immediately follow after *D*" (if *End,* then not *D),* then the violating behavior which is the negation form of the defined rule will be "*End* follow after *D*" (if *End,* then *D).* All violating patterns will be used in matching violations in original event log in the next step of rule checking on log.

Event Process Language Syntax

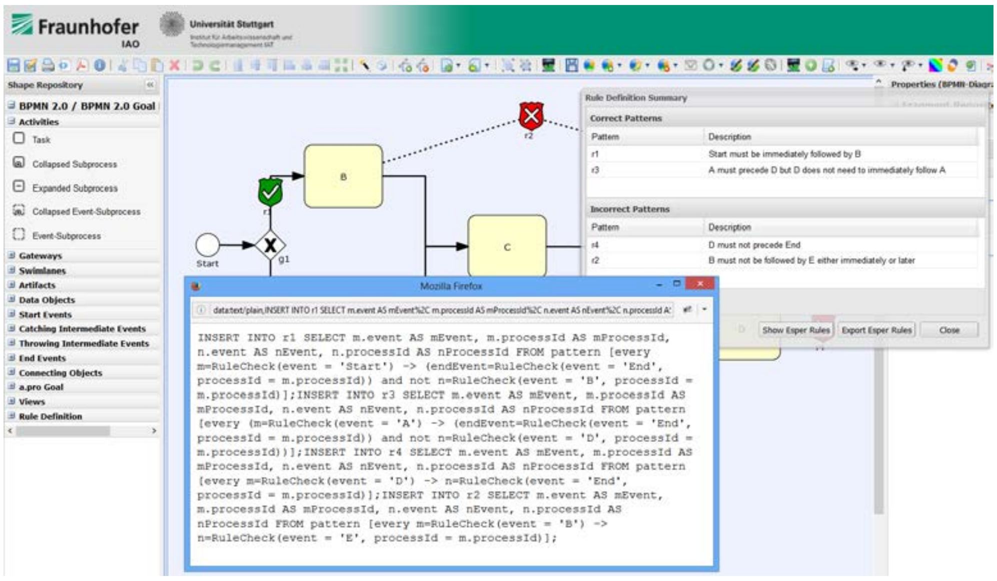
```
INSERT INTO r4 SELECT m.event AS mEvent, m.processId AS mProcessId,
n.event AS nEvent, n.processId AS nProcessId
FROM pattern
[every m=RuleCheck(event = 'D') ->
       n=RuleCheck(event = 'End', processId = m.processId)]
```

The INSERT INTO clause is recast as a means of forwarding events to other streams for further downstream processing. A PATTERN may appear anywhere in the from clause of an EPL statement. The notation of "->" indicates a PATTERN of event ordering. In this case, "D->End" means "*End* immediately follows *D*". This is a violating behavior to be detected in CEP engine. There is a clause of "processId = m.processId", this limits looking in only the predecessor, and successor must be in the process instance.

Table 6.2 Rule creation of graphical rules to Esper rules

Rule r1:	*B* must immediately follow *Start*
	`INSERT INTO r1 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every m=RuleCheck(event = 'Start') -> (endEvent=RuleCheck(event = 'End', processId = m.processId)) and not n=RuleCheck(event = 'B', processId = m.processId)]`
Rule r2:	*E* must not follow *B* either immediately or later
	`INSERT INTO r2 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every m=RuleCheck(event = 'B') -> n=RuleCheck(event = 'E', processId = m.processId)];`
Rule r3:	*A* must precede *D* but *D* does not need to immediately follow *A*
	`INSERT INTO r3 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every (m=RuleCheck(event = 'A') -> (endEvent=RuleCheck(event = 'End', processId = m.processId)) and not n=RuleCheck(event = 'D', processId = m.processId))]`
Rule r4:	*End* must not immediately follow *D*
	`INSERT INTO r4 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every m=RuleCheck(event = 'D') -> n=RuleCheck(event = 'End', processId = m.processId)]`

The Figure 6.8 illustrates the implementation of rule creation in Oryx process editor. All the graphical rules defined on the process model are generated into Esper rules which are shown in separate window and can be exported into a text file for future use.



Rule Definition Summary

Correct Patterns

Pattern	Description
r1	Start must be immediately followed by B
r3	A must precede D but D does not need to immediately follow A

Incorrect Patterns

Pattern	Description
r4	D must not precede End
r2	B must not be followed by E either immediately or later

```
INSERT INTO r1 SELECT m.event AS mEvent, m.processId AS mProcessId,
n.event AS nEvent, n.processId AS nProcessId FROM pattern [every
m=RuleCheck(event = 'Start') -> (endEvent=RuleCheck(event = 'End',
processId = m.processId)) and not n=RuleCheck(event = 'B', processId =
m.processId)];INSERT INTO r3 SELECT m.event AS mEvent, m.processId AS
mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern
[every (m=RuleCheck(event = 'A') -> (endEvent=RuleCheck(event = 'End',
processId = m.processId)) and not n=RuleCheck(event = 'D', processId =
m.processId))];INSERT INTO r4 SELECT m.event AS mEvent, m.processId AS
mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern
[every m=RuleCheck(event = 'D') -> n=RuleCheck(event = 'End',
processId = m.processId)];INSERT INTO r2 SELECT m.event AS mEvent,
m.processId AS mProcessId, n.event AS nEvent, n.processId AS
nProcessId FROM pattern [every m=RuleCheck(event = 'B') ->
n=RuleCheck(event = 'E', processId = m.processId)];
```

Figure 6.8 Implementation of rule creation in Oryx process editor

6.5 Rule Checking on Log

After rule creation, the Esper rules are generated and they are now ready for the compliance checking step. The rule checking on log is implemented in Oryx process editor with Esper Complex Event Processing (CEP) mechanism to analyze event series for detecting violations among events.

Figure 6.9 illustrates the rule checking on log using Esper CEP. The mechanism considers event streams as the source of data instead of normal tables. The event streams are the sequence of event objects which are written as Plan Old Java Object (POJO). In this work, the event streams are the sequence of data from the event log which have been constructed in Java objects. Each java object consists of event name and process id attributes. The event streams flow through the set of defined rules which are registered as listeners in CEP engine. The CEP engine processes each event from the event streams and notifies an event trigger whenever an event data violates to the defined rules.

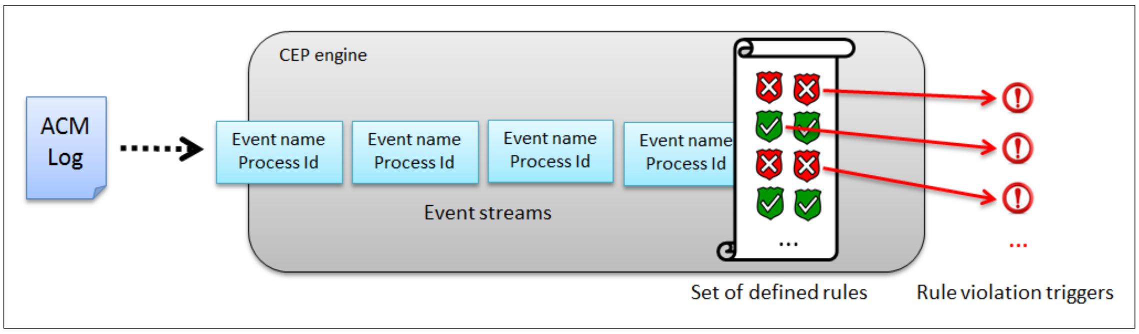


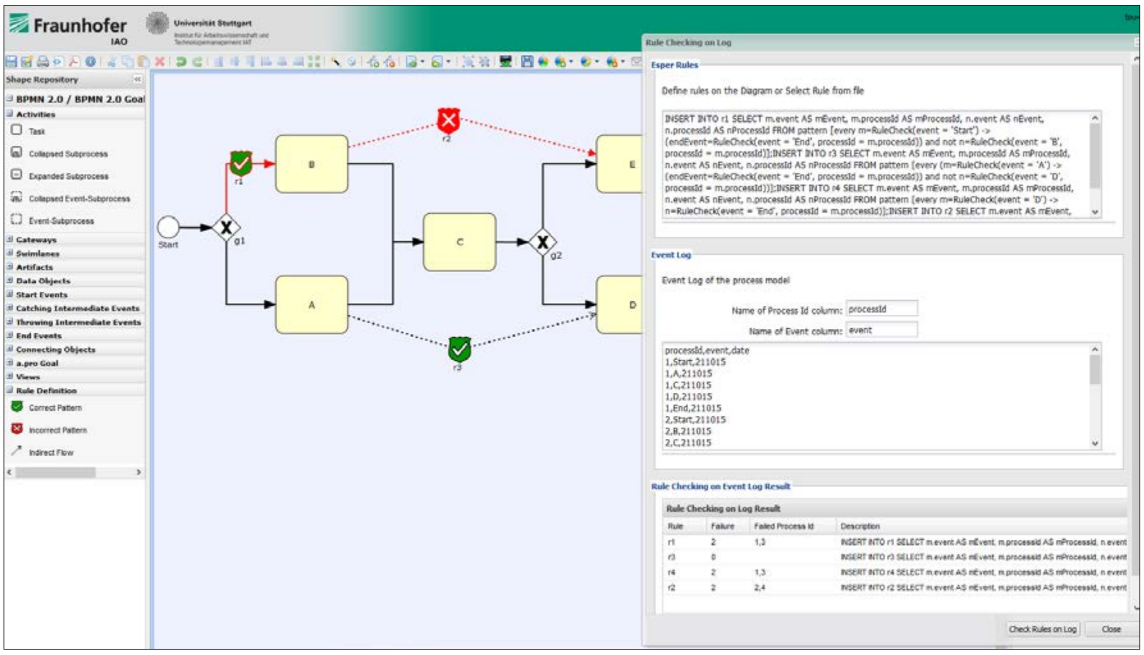
Figure 6.9 Oryx Rule checking on log using Esper Complex Event Processing

Figure 6.10 illustrates the implementation of rule checking on log in Oryx process editor. The rule checking on log panel consists of three sections *Esper rules*, *Event log* and *Rule checking on log result*.

The *Esper rules* section automatically displays Esper rules which are graphically defined on Oryx canvas by calling rule creation function (See 6.4 Rule Creation). However, this section is editable. It allows the editing of the generated rules or manual typing of the Esper rules into the text field.

The *Event log* section is used for importing data from an event log file. The event log data is in form of CSV file and separate with comma (,). The column name of process Id and event have to be declared so that the data from the correct columns can be extracted and used in the rule checking procedure.

Esper rules and *Event log* are fetched into CEP engine and performed the Oryx rule checking on log as demonstrated in Figure 6.9. The Esper rules are registered as listeners in CEP engine and the event log in form of event steams flow through the set of Esper rules listeners, whenever an event data violoates the Esper rules patterns, then CEP engine send out an event trigger which means a rule violation is occurred. All the rule violations or violating behaviors are recorded and visualized graphically on the process model as well as an information table result which is discussed in result visualization section (See 6.6 Result Visualization).



```
INSERT INTO r1 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent,
n.processId AS nProcessId FROM pattern [every m=RuleCheck(event = 'Start') ->
(endEvent=RuleCheck(event = 'End', processId = m.processId)) and not n=RuleCheck(event = 'B',
processId = m.processId)];INSERT INTO r3 SELECT m.event AS mEvent, m.processId AS mProcessId,
n.event AS nEvent, n.processId AS nProcessId FROM pattern [every (m=RuleCheck(event = 'A') ->
(endEvent=RuleCheck(event = 'End', processId = m.processId)) and not n=RuleCheck(event = 'D',
processId = m.processId))];INSERT INTO r4 SELECT m.event AS mEvent, m.processId AS mProcessId,
n.event AS nEvent, n.processId AS nProcessId FROM pattern [every m=RuleCheck(event = 'D') ->
n=RuleCheck(event = 'End', processId = m.processId)];INSERT INTO r2 SELECT m.event AS mEvent,
```

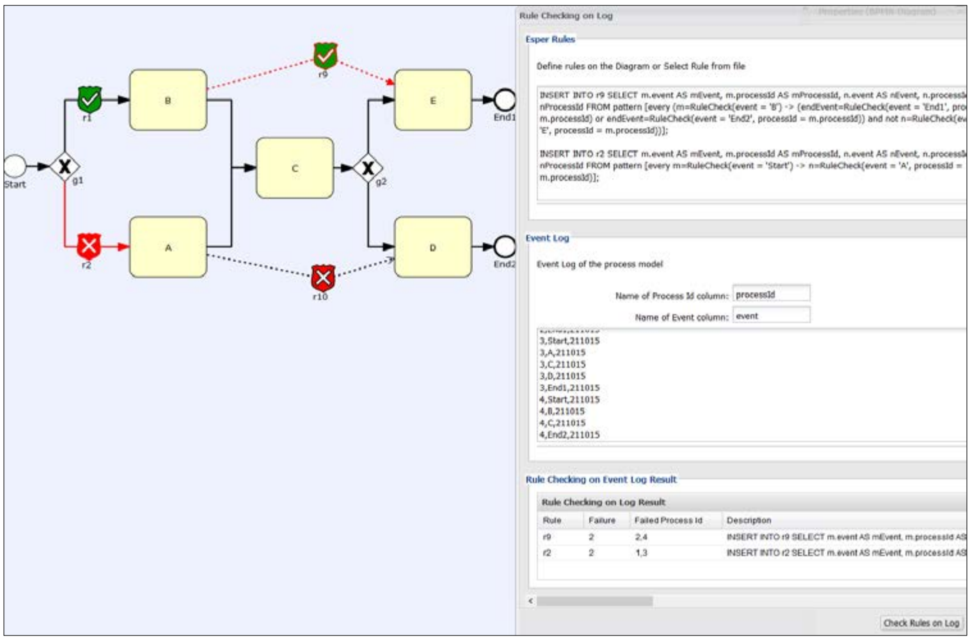
```
processId,event,date
1,Start,211015
1,A,211015
1,C,211015
1,D,211015
1,End,211015
2,Start,211015
2,B,211015
2,C,211015
```

Rule Checking on Log Result

Rule	Failure	Failed Process Id	Description
r1	2	1,3	INSERT INTO r1 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event
r3	0		INSERT INTO r3 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event
r4	2	1,3	INSERT INTO r4 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event
r2	2	2,4	INSERT INTO r2 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event

Figure 6.10 Implementation of rule checking on log in Oryx process editor

The rule checking on log implemented in Oryx process editor is able to be used with process models which have multiple ends. An example of a multiple ends process model is showed in Figure 6.11. The rule creation generates additional clauses for multiple end events. Rule checking on log applies the same mechanism as illustrated in Figure 6.9. Even thought there is more than one end, the algorithm of rule checking on log is also suitable for this compliance checking condition.



Rule Checking on Log

Esper Rules

Define rules on the Diagram or Select Rule from file

```
INSERT INTO r9 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.process
nProcessId FROM pattern [every (m=RuleCheck(event = 'B') -> (endEvent=RuleCheck(event = 'End1', pro
m.processId) or endEvent=RuleCheck(event = 'End2', processId = m.processId)) and not n=RuleCheck(ev
'E', processId = m.processId))];

INSERT INTO r2 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.process
nProcessId FROM pattern [every m=RuleCheck(event = 'Start') -> n=RuleCheck(event = 'A', processId =
m.processId)];
```

Event Log

Event Log of the process model

Name of Process Id column: processId

Name of Event column: event

```
3,Start,211015
3,A,211015
3,C,211015
3,D,211015
3,End1,211015
4,Start,211015
4,B,211015
4,C,211015
4,End2,211015
```

Rule Checking on Event Log Result

Rule Checking on Log Result

Rule	Failure	Failed Process Id	Description
r9	2	2,4	INSERT INTO r9 SELECT m.event AS mEvent, m.processId AS
r2	2	1,3	INSERT INTO r2 SELECT m.event AS mEvent, m.processId AS

Check Rules on Log

Figure 6.11 Rule checking in Oryx process editor for multiple-ends processes

6.6 Result Visualization

During processing the rule checking on log, whenever an event data violates the defined rules, the CEP engines send out an event trigger which means a rule violation has occurred and the event trigger information is recorded including rules and process instance identifications for use in result visualization.

In this work, there are two perspectives of the result visualization including graphical result in Oryx canvas and information table result. The graphical result in Oryx canvas provides an overview of the violations on the process model by highlighting the violations paths, while the table result gives insight information about rules, number of violations and process instances.

6.6.1 Graphical Result

The graphical result in Oryx canvas represents the violations by highlighting the violation paths in red. The graphical result provides a clear view of where the violations occurred in the overall process. As shown in Figure 6.12, rule *r1*, *r2* and *r4* are marked in red but *r3* is not. This mean there are some violations or misbehaviors occurring against the rule *r1*, *r2* and *r4* but there is no violation occurring against rule r3. The details are explained in Table 6.3.

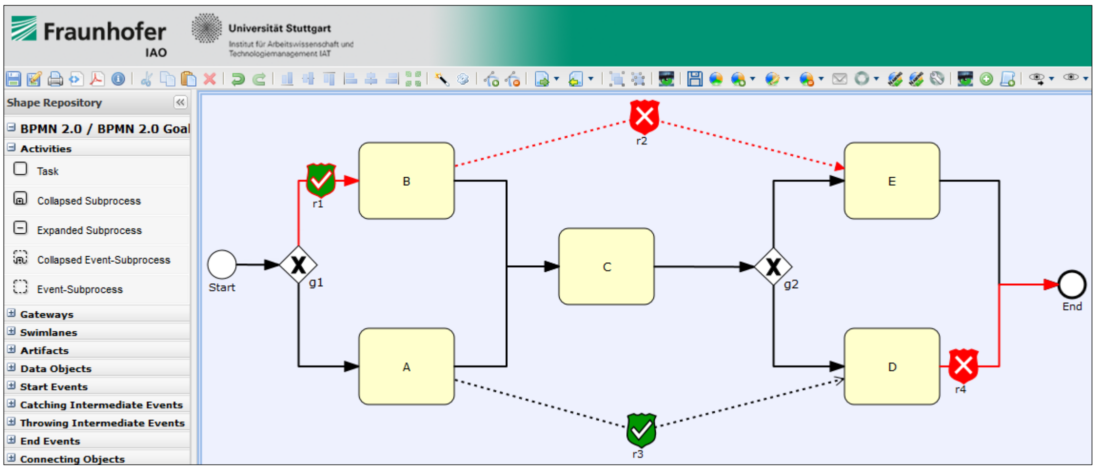


Figure 6.12 Graphical result of rule checking on log displayed in Oryx canvas

Table 6.3 Rule checking on log result explanations

Defined rules	Violation	Description
Rule r1: *B* must immediately follow *Start*	Yes	there is some process instances which *B* does not follow *Start*
Rule r2: *E* must not follow *B* either immediately or later	Yes	there is some process instances which *E* follow *B*
Rule r3: *A* must precede *D* but *D* does not need to immediately follow A	No	True for all process instances
Rule r4: *End* must not immediately follow *D*	Yes	there is some process instances which *End* immediately follow *D*

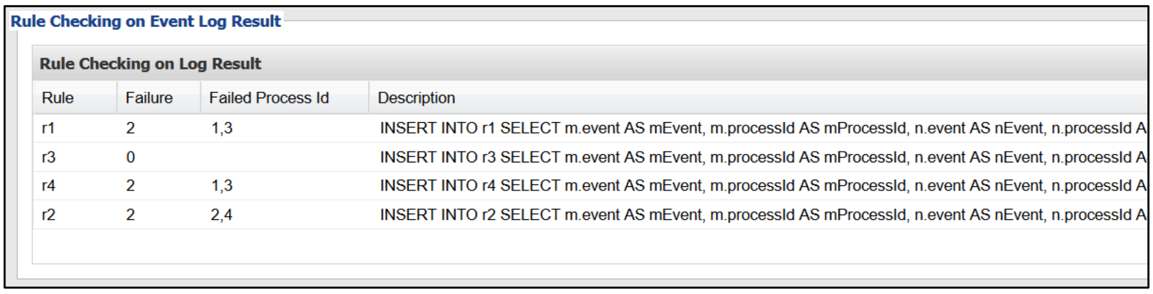
6.6.2 Information Table Result

The graphical result reveals the violations on the overall process, however, the detailed information perspective makes more understanding about the violations. In this work, the information table result of rule checking on log is implemented so that the insight information about rules and process instance can be clarified.

The information table result gives the result of rule checking on log which can answer these four questions:

Questions	Answers
1. Which rule has violations?	rule *r1* has *2* violations which occurred in process instance *1* and *3*
2. How many violations against the rule?	rule *r2* has *2* violations which occurred in process instance *2* and *4*
3. Which process instance has violating behavior against the rule?	rule *r3* has no violations
4. Which rule statement has been used?	rule *r4* has *2* violations which occurred in process instance *1* and *3*

Figure 6.13 is the rule checking on log result of graphical rule definition shown in Figure 6.7. The result can be read as:


Rule Checking on Event Log Result

Rule Checking on Log Result

Rule	Failure	Failed Process Id	Description
r1	2	1,3	INSERT INTO r1 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId A
r3	0		INSERT INTO r3 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId A
r4	2	1,3	INSERT INTO r4 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId A
r2	2	2,4	INSERT INTO r2 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId A

Figure 6.13 The table result of rule checking on log in Oryx process editor

Next chapter, real-life data is used for evaluating the implemented prototype solution of the proposed approach.

7 Evaluation

In chapter 6, the prototype solution is implemented for proving the approach of compliance checking in unstructured processes using process mining.

In this chapter, the implemented prototype is evaluated with a real-life dataset. The evolution serves the purposes including verifying correctness of the approach and examining whether the developed approach is applicable for real-life event logs.

7.1 Source of Real World Dataset

The proposed approach in this work is compliance checking for unstructured processes using process mining. An unstructured process is considered as a real-life event log produced by knowledge workers performing their works in unpredictable environment. Therefore case-by-case investigation is needed in such situations, for example insurance claims.

In this work, a real world dataset from a car insurance claim system is used. The claiming system is ARPOS developed by Fraunhofer IAO [69]. ARPOS stands for Automatic Rule-based Process control for Online claims processing (Automatische Regelbasierte Prozessteuerung zur Onlineabwicklung von Schadensfällen). ARPOS is solution for processing car damage claims for insurance companies. The solution consists of several software components that perform over a service process. The purpose of this solution is checking and analyzing incoming car claims from various sources then generating reports for insurance companies.

The service of ARPOS starts from receiving a car claim. Then various checks are carried out paralleled with extracting images of the claim case. Afterwards, the automatic examination proceeds on the receiving claim. The claim may be completed or required further analysis by claiming experts for specialized investigations. An investigating report is then generated and sent back to the insurance company.

7.2 Log Preparation

In this master thesis, a real-life data log is extracted from the ARPOS system. The considered period of dataset is around one year (from 17 September 2015 until 2 August 2016).

The ARPOS system is used for car claims investigation. The logs produced by ARPOS process executions capture activities in claiming processes including automatic examinations and expert investigations. ARPOS generates event logs in XML which describes properties of each case. However, the format of the event logs is designed for specific purpose. Therefore, to be able to evaluate the prototype solution by the real-life event logs from ARPOS, the ETL process is employed for event log preparation.

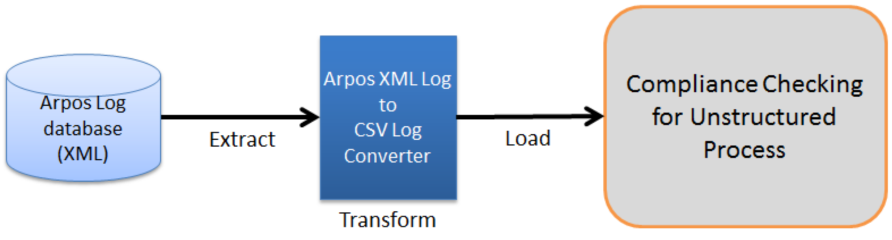

Arpos Log database (XML)
Extract
Arpos XML Log to CSV Log Converter
Transform
Load
Compliance Checking for Unstructured Process

Figure 7.1 Log preparation

The log preparation from ARPOS to the compliance checking in unstructured processes approach is illustrated in Figure 7.1. The logs from ARPOS database are extracted and fetch into transformation function. Due to difference of event logs format, a converter is developed for the purpose of transforming ARPOS XML logs into CSV format. The ARPOS event logs consist of various attributes for each case, however the needed attributes are case identification, event name, and timestamp. After extracting needed attributes and transforming in CSV format, the ARPOS in CSV format is loaded to the prototype solution for evaluating the proposed approach.

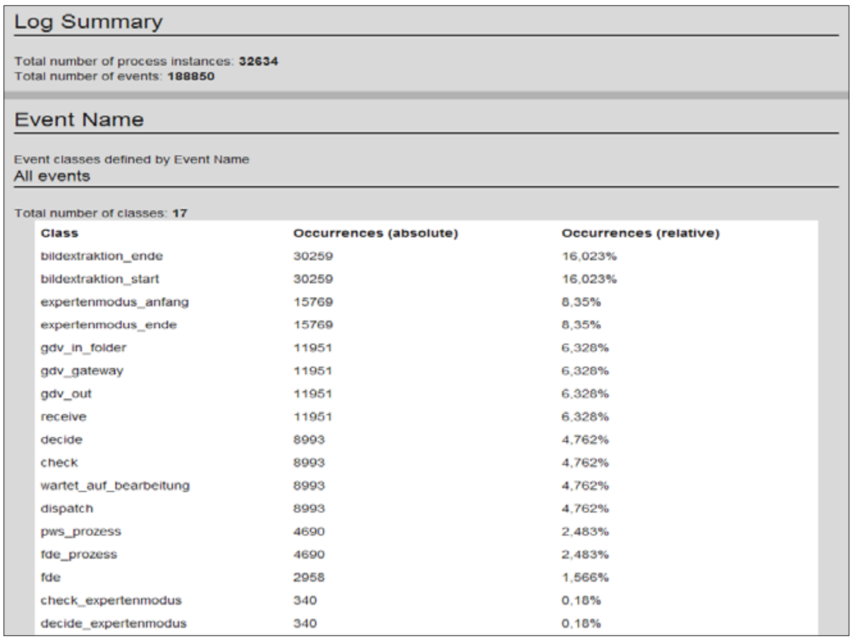
7.3 Prototype Evaluation

The dataset used to evaluate the prototype is the real-life logs which are generated from ARPOS system. The logs have been transformed into CSV format which is able to be fetched into the prototype (See section 7.2). The evaluations of each step of the approach are discussed in the following.

7.3.1 Process Mining in PROM

The first step of the approach is transforming less structured into more structured processes by a process mining technique. In this work, PROM version 6.5.1 is used for performing process mining. The chosen mining algorithm is heuristic miner which is able to construct the main behaviors from a given real-life log (See section 5.1.2 and 6.2).

After ARPOS log preparation, the CSV log is given to PROM. A CSV to XES plugin converts CSV log into XES. The log summary shown in Figure 7.2 is displayed after converting into XES. The log summary illustrates that the ARPOS log has 17 event types, 32,634 process instances and 188,850 executed events. It means that over one-year period, ARPOS has investigated 32,634 cases of car claims, 17 task types have been used in investigating processes and 188,850 tasks in total have been executed.


Log Summary

Total number of process instances: **32634**
Total number of events: **188850**

Event Name

Event classes defined by Event Name
All events

Total number of classes: **17**

Class	Occurrences (absolute)	Occurrences (relative)
bildextraktion_ende	30259	16,023%
bildextraktion_start	30259	16,023%
expertenmodus_anfang	15769	8,35%
expertenmodus_ende	15769	8,35%
gdv_in_folder	11951	6,328%
gdv_gateway	11951	6,328%
gdv_out	11951	6,328%
receive	11951	6,328%
decide	8993	4,762%
check	8993	4,762%
wartet_auf_bearbeitung	8993	4,762%
dispatch	8993	4,762%
pws_prozess	4690	2,483%
fde_prozess	4690	2,483%
fde	2958	1,566%
check_expertenmodus	340	0,18%
decide_expertenmodus	340	0,18%

Figure 7.2 Log Summary generated by PROM

Next, heuristic miner is applied to discover a process model from the log. The configurable dependency parameter of this algorithm is used to measure relationship between activities. A high value denotes a strong dependency, while a low value denotes a weak dependency. In this case, the main behaviors of the given real-life log is focused therefore high dependency values is considered. Several dependency values are experimented combined with interviewing with the system expert to review the output process models. The process chosen by the expert is the process with selected dependency value 100 as it was judged to be the most realistic process model for the claiming system.

Heuristic miner discovers first a heuristic net, and then the heuristic net is converted into Petri net. Lastly, Petri net is transformed into BPMN in order to be imported into Fraunhofer IAO Oryx process editor.

The result diagram is shown in Figure 7.3 (left). The heuristics miner in PROM constructs a process model from an event log over one-year period relatively fast. It can generate a well-organized output model with a nice curve sequence flows even the input is given from a complex system. However, the result process model has some issues as follows:

- The Start notation is created but it does not connect to the diagram
- One outgoing edge of every event is forwarded to End notation

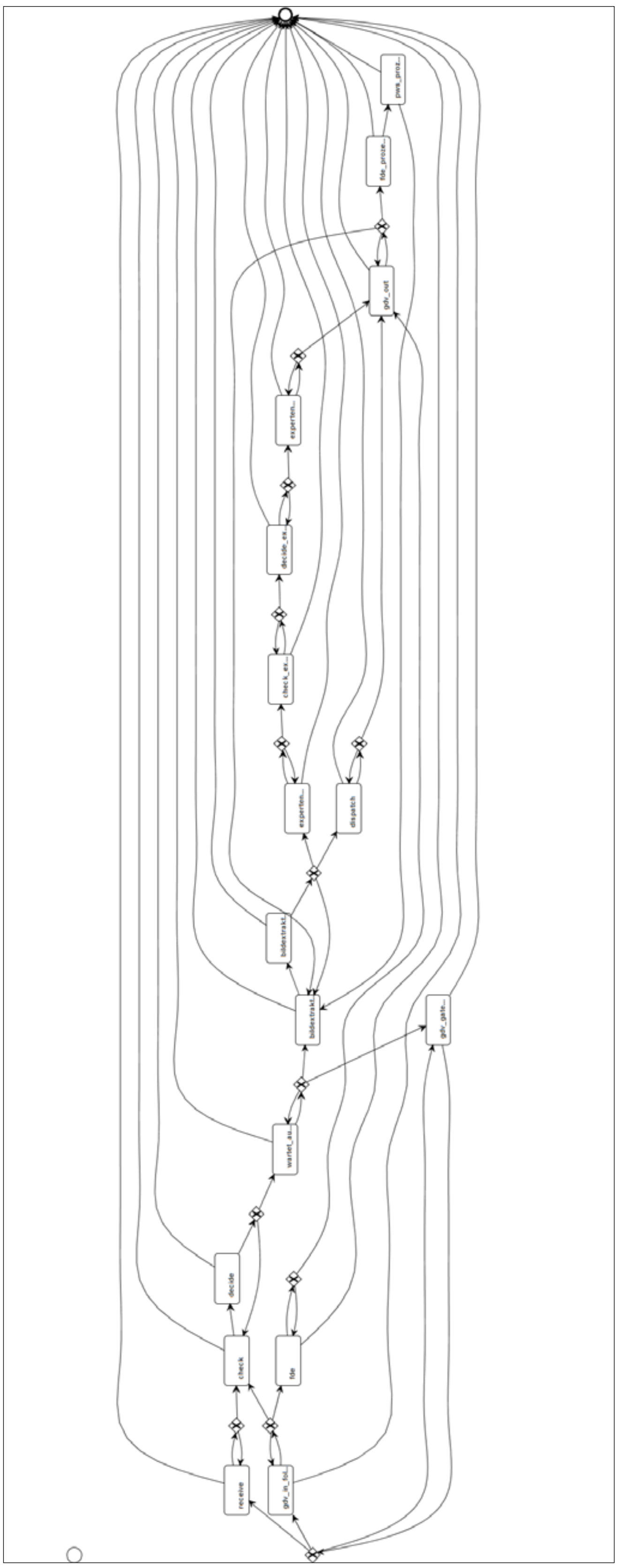


Figure 7.3 The discovered process model by PROM

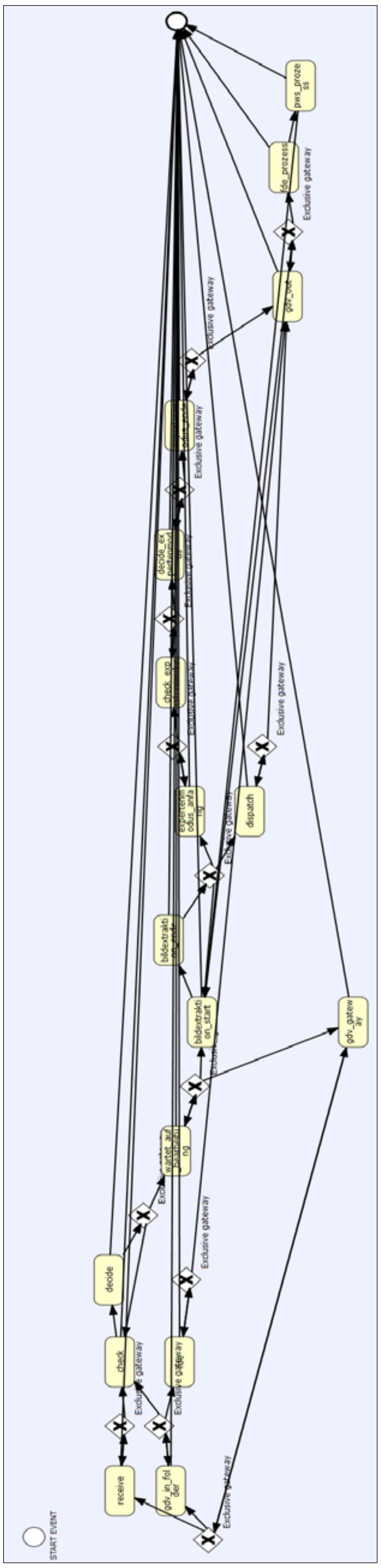


Figure 7.4 The imported process model to Oryx process editor

7.3.2 Oryx Process Editor

7.3.2.1 Process Model Importation

After obtaining the discovered process model (See Figure 7.3), the process model is imported into Oryx process editor. Before importing, however, the BPMN converter is applied to convert PROM BPMN into Oryx BPMN. After converting, the process model is imported by Oryx existing import functionality. The imported process model displaying in Oryx canvas is shown in Figure 7.4. However, there is a limitation of displaying a complex process in Oryx.

- Due to lack of rendering curve sequence flows in Oryx, representing a complex process model can be complicated and not understandable. Therefore a manual diagram adjustment is needed.

7.3.2.2 Diagram Adjustment

Due to occurring issues in PROM and Oryx mentioned in section 7.3.1 and 7.3.2.1, the imported model has to be refined before further evaluating. The manual adjustments which are applied on the imported diagram (Figure 7.4) listed in the following:

i. The Start notation is created but it does not connect to the diagram
   - Adjustment: Connecting the Start notation to the diagram

ii. One outgoing edge of every executed event is forwarded to End notation
   - Adjustment: the exceeded sequence flows

iii. Lack of rendering curve sequence flows in Oryx
   - Adjustment: Aligning sequence flows manually

The process model which is already adjusted shown in Figure 7.5. The process model is now more understandable and can be used in the next step.

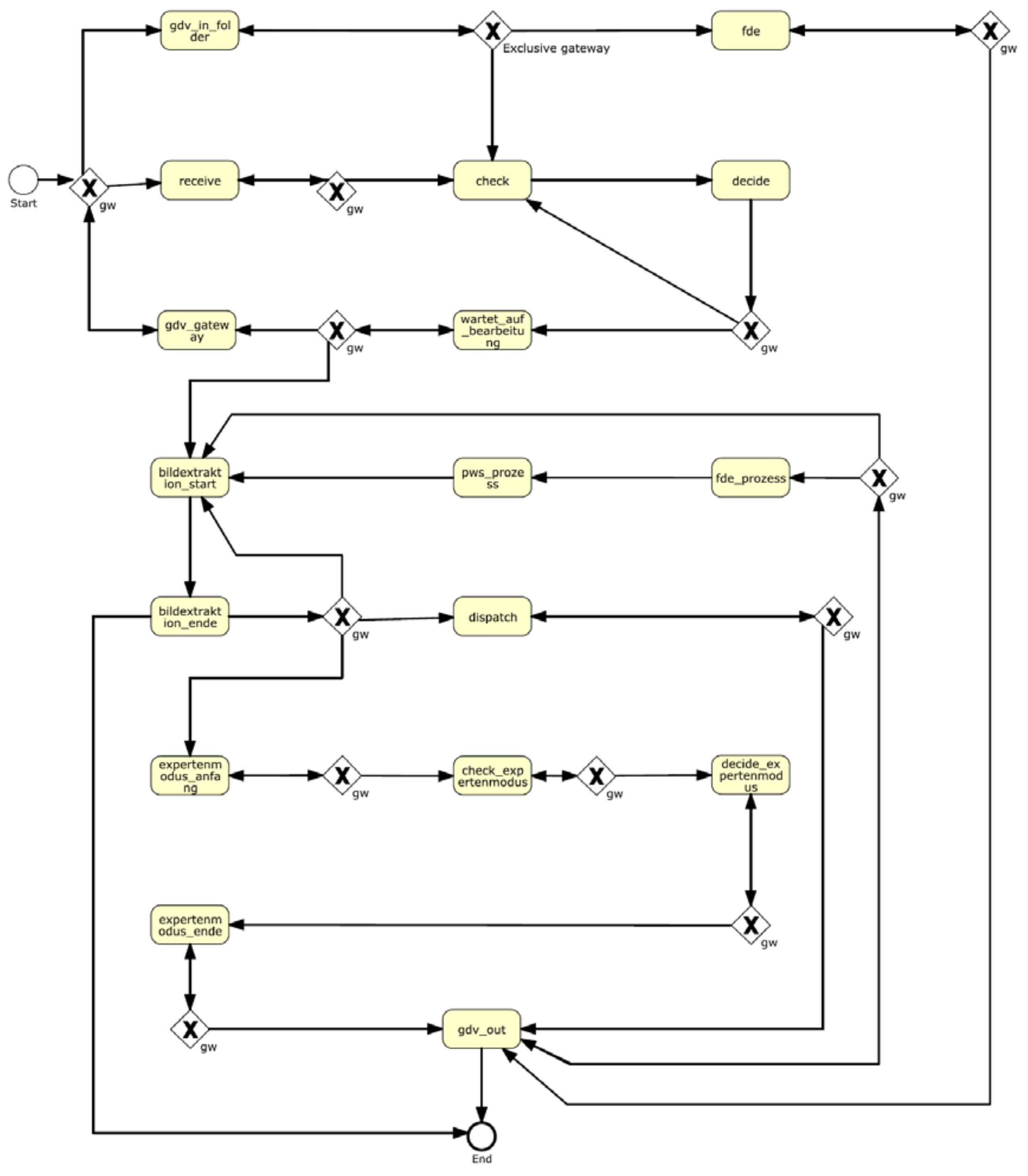


Figure 7.5 The adjusted ARPOS process model

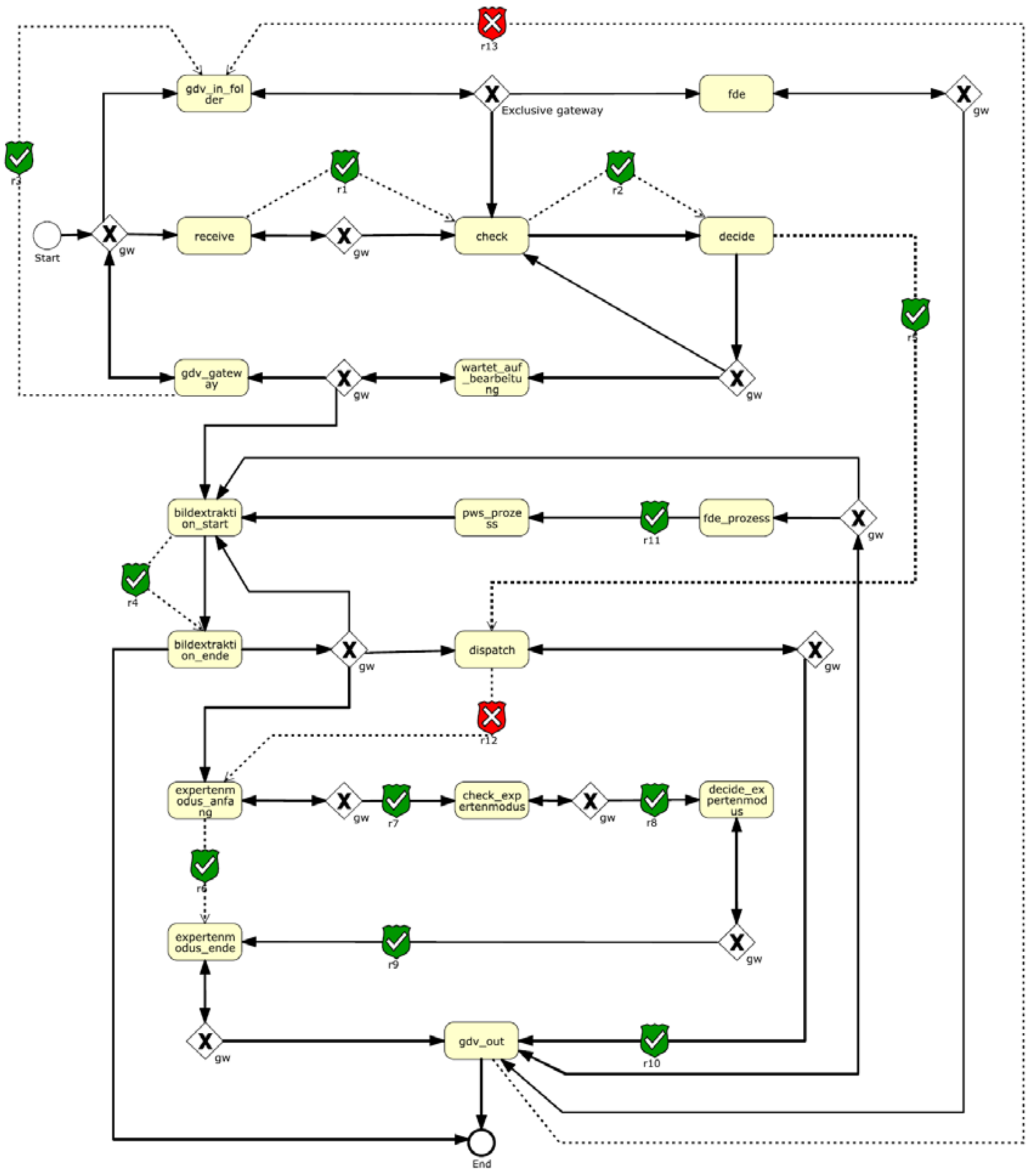
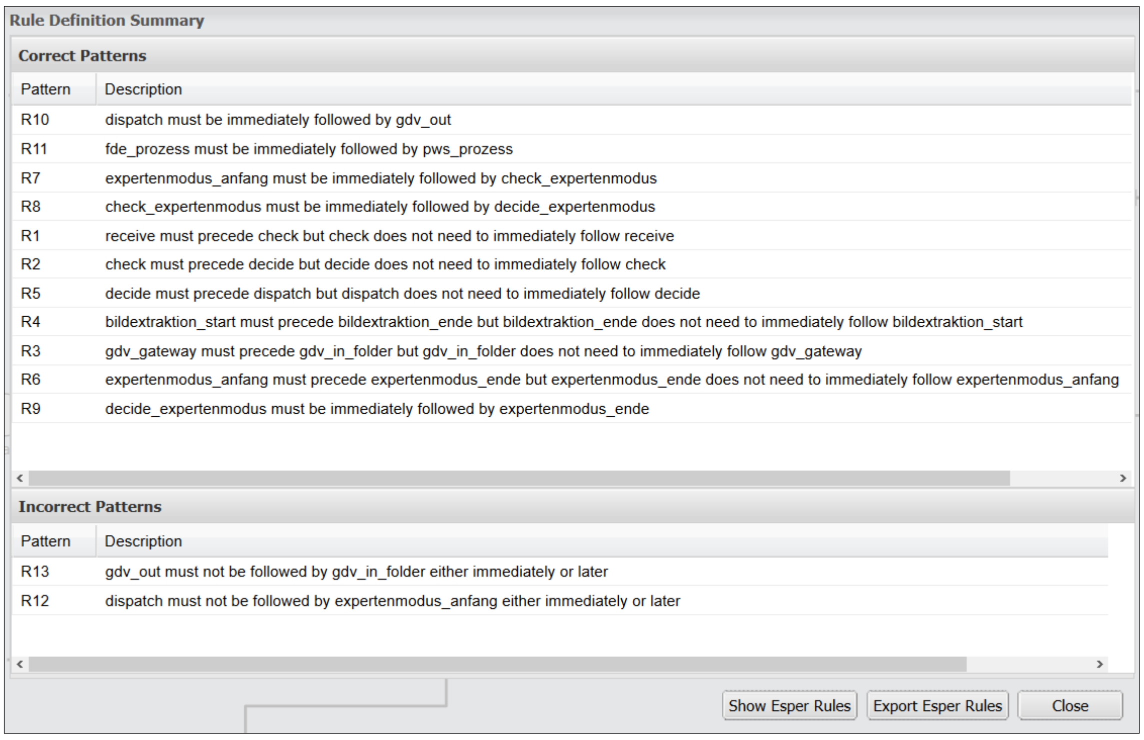


Figure 7.6 ARPOS process model with graphical rule definiton

7.3.2.3 Rule Definition

The stencil set for defining rules consists of *Correct pattern*, *Incorrect pattern,* and *Indirect flow* notations (See section 6.3.2). Rules are defined by drag-and-drop on the aligned process model. In this evaluation, the process expert defines 13 rules. There are 11 correct rule patterns and 2 incorrect rule patterns. The process model with defined graphical rules is demonstrated in Figure 7.6. The list of defined rules and their descriptions are shown in Figure 7.2.

Additionally, the rule definition summary function shows the overview of all the defined rules on a diagram and their rule descriptions. The rule summary function correctly generated a list of 13 defined rules and interprets rule descriptions. The rule summary is illustrated in Figure 7.7.


Rule Definition Summary

Correct Patterns

Pattern	Description
R10	dispatch must be immediately followed by gdv_out
R11	fde_prozess must be immediately followed by pws_prozess
R7	expertenmodus_anfang must be immediately followed by check_expertenmodus
R8	check_expertenmodus must be immediately followed by decide_expertenmodus
R1	receive must precede check but check does not need to immediately follow receive
R2	check must precede decide but decide does not need to immediately follow check
R5	decide must precede dispatch but dispatch does not need to immediately follow decide
R4	bildextraktion_start must precede bildextraktion_ende but bildextraktion_ende does not need to immediately follow bildextraktion_start
R3	gdv_gateway must precede gdv_in_folder but gdv_in_folder does not need to immediately follow gdv_gateway
R6	expertenmodus_anfang must precede expertenmodus_ende but expertenmodus_ende does not need to immediately follow expertenmodus_anfang
R9	decide_expertenmodus must be immediately followed by expertenmodus_ende

Incorrect Patterns

Pattern	Description
R13	gdv_out must not be followed by gdv_in_folder either immediately or later
R12	dispatch must not be followed by expertenmodus_anfang either immediately or later

Figure 7.7 Rule definition summary

7.3.2.4 Rule Creation

Rule creation function interprets graphical rules into rule statements which are expressed in Event Process Language (EPL). The rule statements are deployed in CEP engine for compliance checking in the next step. All 13 defined rules from previous step have been correctly interpreted. Table 7.1 demonstrates rule statements which are interpreted from the graphical defined rules by rule creation function.

Table 7.1 Rule statements interpreted from the defined graphical rules

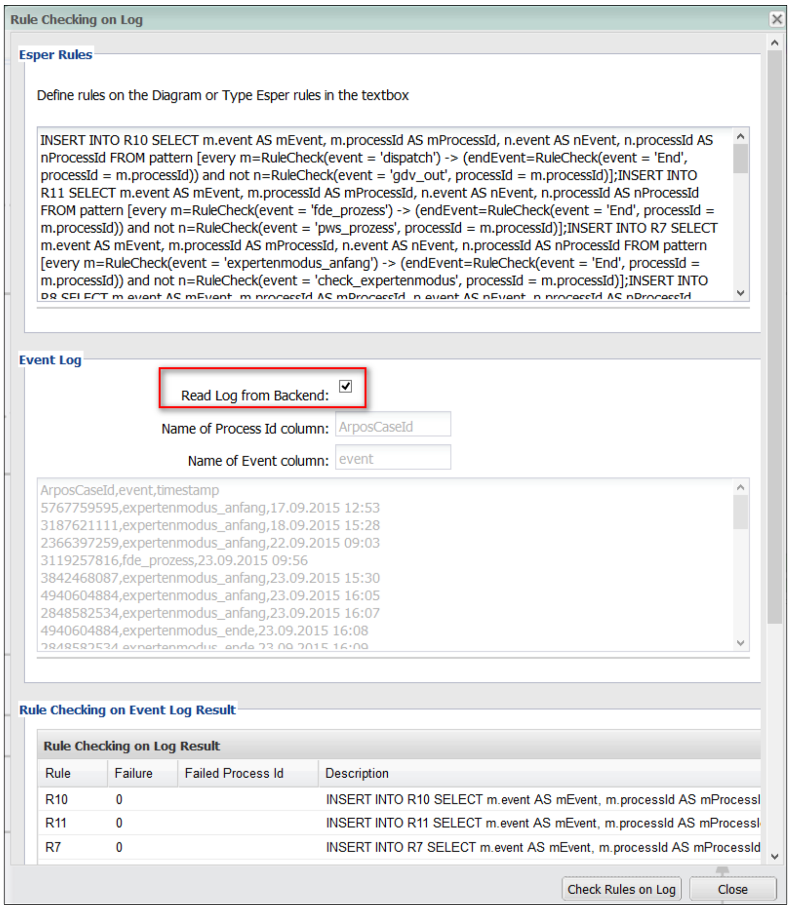
Rule	Rule statement
R1	`INSERT INTO R1 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every (m=RuleCheck(event = 'receive') -> (endEvent=RuleCheck(event = 'End', processId = m.processId)) and not n=RuleCheck(event = 'check', processId = m.processId))];`
R2	`INSERT INTO R2 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every (m=RuleCheck(event = 'check') -> (endEvent=RuleCheck(event = 'End', processId = m.processId)) and not n=RuleCheck(event = 'decide', processId = m.processId))];`
R3	`INSERT INTO R3 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every (m=RuleCheck(event = 'gdv_gateway') -> (endEvent=RuleCheck(event = 'End', processId = m.processId)) and not n=RuleCheck(event = 'gdv_in_folder', processId = m.processId))];`
R4	`INSERT INTO R4 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every (m=RuleCheck(event = 'bildextraktion_start') -> (endEvent=RuleCheck(event = 'End', processId = m.processId)) and not n=RuleCheck(event = 'bildextraktion_ende', processId = m.processId))];`
R5	`INSERT INTO R5 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every (m=RuleCheck(event = 'decide') -> (endEvent=RuleCheck(event = 'End', processId = m.processId)) and not n=RuleCheck(event = 'dispatch', processId = m.processId))];`

R6	INSERT INTO R6 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every (m=RuleCheck(event = 'expertenmodus_anfang') -> (endEvent=RuleCheck(event = 'End', processId = m.processId)) and not n=RuleCheck(event = 'expertenmodus_ende', processId = m.processId))];
R7	INSERT INTO R7 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FR OM pattern [every m=RuleCheck(event = 'expertenmodus_anfang') -> (endEvent=RuleCheck(event = 'End', processId = m.processId)) and not n=RuleCheck(event = 'check_expertenmodus', processId = m.processId)];
R8	INSERT INTO R8 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every m=RuleCheck(event = 'check_expertenmodus') -> (endEvent=RuleCheck(event = 'End', processId = m.processId)) and not n=RuleCheck(event = 'decide_expertenmodus', processId = m.processId)];
R9	INSERT INTO R9 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every m=RuleCheck(event = 'decide_expertenmodus') -> (endEvent=RuleCheck(event = 'End', processId = m.processId)) and not n=RuleCheck(event = 'expertenmodus_ende', processId = m.processId)];
R10	INSERT INTO R10 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every m=RuleCheck(event = 'dispatch') -> (endEvent=RuleCheck(event = 'End', processId = m.processId)) and not n=RuleCheck(event = 'gdv_out', processId = m.processId)];
R11	INSERT INTO R11 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every m=RuleCheck(event = 'fde_prozess') -> (endEvent=RuleCheck(event = 'End', processId = m.processId)) and not n=RuleCheck(event = 'pws_prozess', processId = m.processId)];
R12	INSERT INTO R12 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every m=RuleCheck(event = 'dispatch') -> n=RuleCheck(event = 'expertenmodus_anfang', processId = m.processId)];
R13	INSERT INTO R13 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern [every m=RuleCheck(event = 'gdv_out') -> n=RuleCheck(event = 'gdv_in_folder', processId = m.processId)];

7.3.2.5 Rule Checking on Log

In this work, compliance checking is performed by checking defined rules on the original log. The rule checking on log panel consists of Esper rules which are generated by rule creation (See section 7.3.2.4 Rule Creation), the section for filling event logs and rule checking on logs result (See section 7.3.2.6)

The 13 rules are graphically defined (See section 7.3.2.3). The real world data log contains anonymized data from the ARPOS system for claims investigations. Over one-year period, ARPOS has investigated 32,634 cases of car claims and 188,850 tasks in total have been executed. However, the developed prototype could support approximately 27,000 executed tasks due to limitation of log size sent to server via browser. Therefore, the prototype is further modified to be able to support a large log by reading the log and manipulating on backend. As the result, the prototype can support all the executed tasks. The modified rule checking on log panel is illustrated in Figure 7.8 and the result of rule checking is demonstrated in Figure 7.9.



Define rules on the Diagram or Type Esper rules in the textbox

```
INSERT INTO R10 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS
nProcessId FROM pattern [every m=RuleCheck(event = 'dispatch') -> (endEvent=RuleCheck(event = 'End',
processId = m.processId)) and not n=RuleCheck(event = 'gdv_out', processId = m.processId)];INSERT INTO
R11 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId
FROM pattern [every m=RuleCheck(event = 'fde_prozess') -> (endEvent=RuleCheck(event = 'End', processId =
m.processId)) and not n=RuleCheck(event = 'pws_prozess', processId = m.processId)];INSERT INTO R7 SELECT
m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId FROM pattern
[every m=RuleCheck(event = 'expertenmodus_anfang') -> (endEvent=RuleCheck(event = 'End', processId =
m.processId)) and not n=RuleCheck(event = 'check_expertenmodus', processId = m.processId)];INSERT INTO
R8 SELECT m.event AS mEvent, m.processId AS mProcessId, n.event AS nEvent, n.processId AS nProcessId
```

Read Log from Backend: ✓

Name of Process Id column: ArposCaseId

Name of Event column: event

```
ArposCaseId,event,timestamp
5767759595,expertenmodus_anfang,17.09.2015 12:53
3187621111,expertenmodus_anfang,18.09.2015 15:28
2366397259,expertenmodus_anfang,22.09.2015 09:03
3119257816,fde_prozess,23.09.2015 09:56
3842468087,expertenmodus_anfang,23.09.2015 15:30
4940604884,expertenmodus_anfang,23.09.2015 16:05
2848582534,expertenmodus_anfang,23.09.2015 16:07
4940604884,expertenmodus_ende,23.09.2015 16:08
2848582534,expertenmodus_ende,23.09.2015 16:09
```

Rule Checking on Log Result

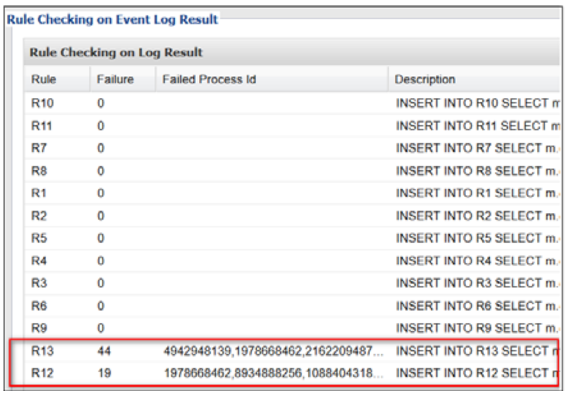
Rule	Failure	Failed Process Id	Description
R10	0		INSERT INTO R10 SELECT m.event AS mEvent, m.processId AS mProcessI
R11	0		INSERT INTO R11 SELECT m.event AS mEvent, m.processId AS mProcessI
R7	0		INSERT INTO R7 SELECT m.event AS mEvent, m.processId AS mProcessId

Figure 7.8 The modified rule checking on log panel

7.3.2.6 Result Visualization

After performing rule checking on log function, the results from validating the defined rule are visualized. The result visualization is represented in two perspectives including graphical result and information table result.

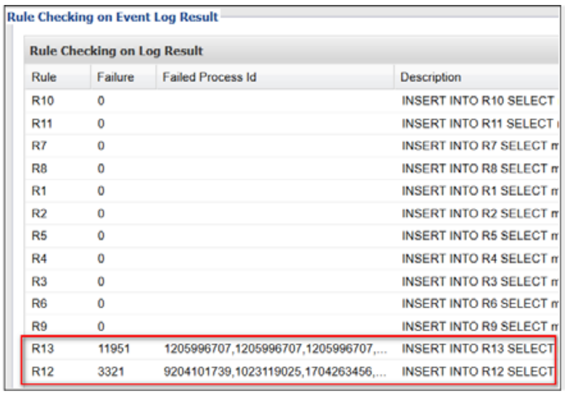
The graphical result is shown in Figure 7.10. The result illustrates ARPOS process model with the defined graphical rules. The rules with violations are shown in red colour. The rules without red marking means there is no violating behaviours against them.



Rule Checking on Event Log Result

Rule Checking on Log Result

Rule	Failure	Failed Process Id	Description
R10	0		INSERT INTO R10 SELECT m
R11	0		INSERT INTO R11 SELECT m
R7	0		INSERT INTO R7 SELECT m.
R8	0		INSERT INTO R8 SELECT m.
R1	0		INSERT INTO R1 SELECT m.
R2	0		INSERT INTO R2 SELECT m.
R5	0		INSERT INTO R5 SELECT m.
R4	0		INSERT INTO R4 SELECT m.
R3	0		INSERT INTO R3 SELECT m.
R6	0		INSERT INTO R6 SELECT m.
R9	0		INSERT INTO R9 SELECT m.
R13	44	4942948139,1978668462,2162209487...	INSERT INTO R13 SELECT m
R12	19	1978668462,8934888256,1088404318...	INSERT INTO R12 SELECT m



Rule Checking on Event Log Result

Rule Checking on Log Result

Rule	Failure	Failed Process Id	Description
R10	0		INSERT INTO R10 SELECT
R11	0		INSERT INTO R11 SELECT
R7	0		INSERT INTO R7 SELECT m
R8	0		INSERT INTO R8 SELECT m
R1	0		INSERT INTO R1 SELECT m
R2	0		INSERT INTO R2 SELECT m
R5	0		INSERT INTO R5 SELECT m
R4	0		INSERT INTO R4 SELECT m
R3	0		INSERT INTO R3 SELECT m
R6	0		INSERT INTO R6 SELECT m
R9	0		INSERT INTO R9 SELECT m
R13	11951	1205996707,1205996707,1205996707,...	INSERT INTO R13 SELECT
R12	3321	9204101739,1023119025,1704263456,...	INSERT INTO R12 SELECT

Figure 7.9 The information table result of rule checking on ARPOS log with 27,000 executed tasks (left) and 188,850 executed tasks (the whole log over one year) (right)

Figure 7.9 shows the information tables result which provides more details about violations. All rules comply with the defined rules except the rule R12 and R13 are violated. The violations are detected in process instances shown in column *Failed Process Id.* In the case of 27,000 executed tasks log, there are 19 and 44 violations occurring against rule R12 and R13 respectively. By processing the whole log, the similar result is produced. Rule R12 and R13 are violated with a larger number of violating process instances. The violating details are explained in section 7.3.2.7 Result Investigation.

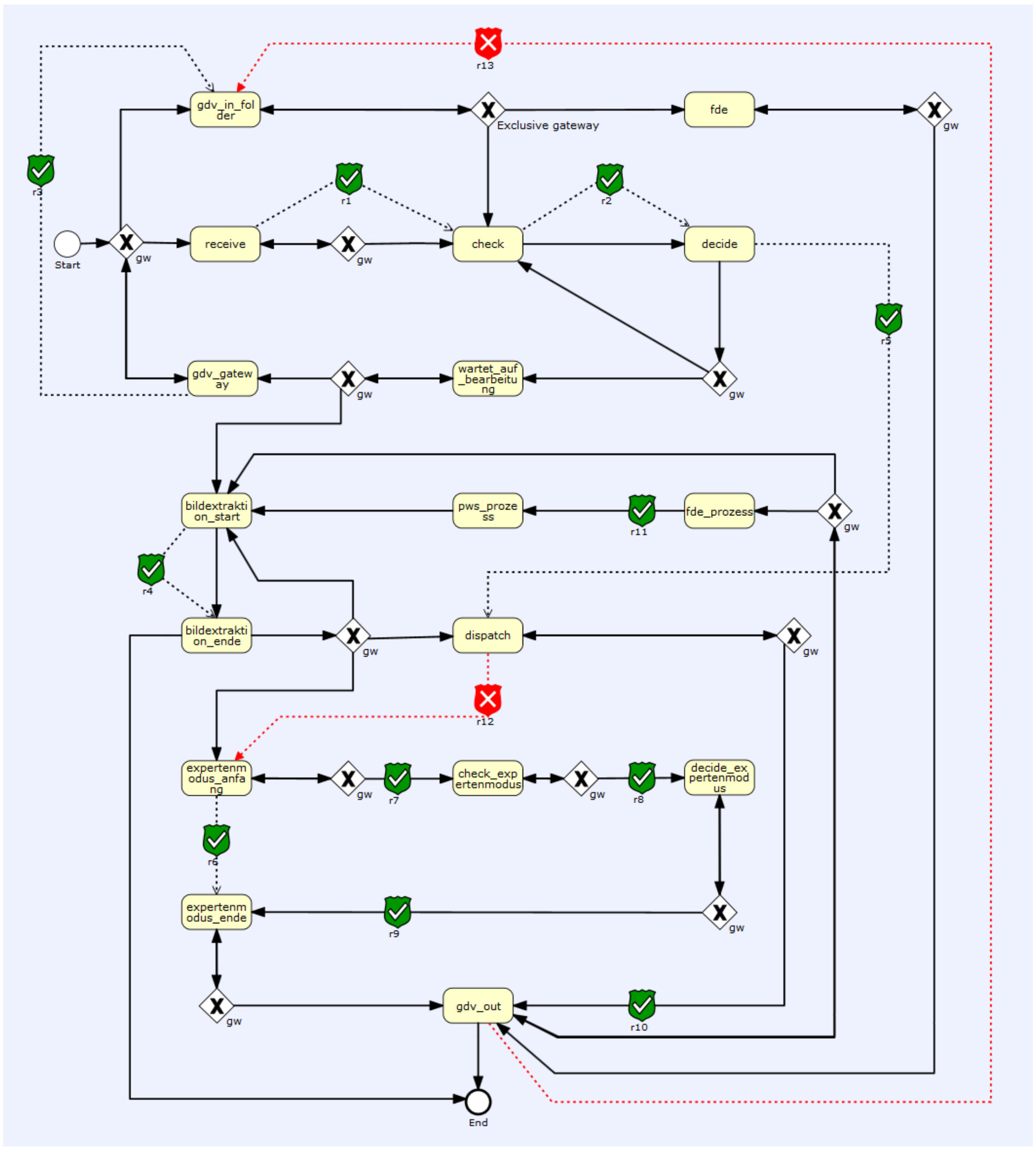
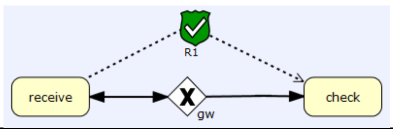
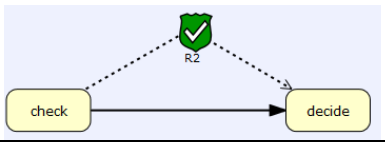
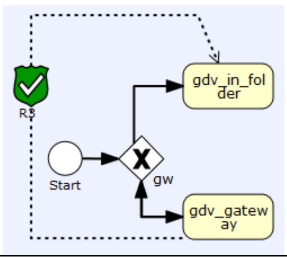
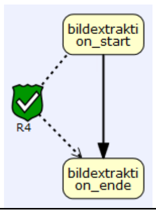
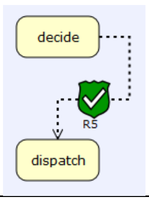
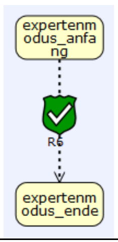
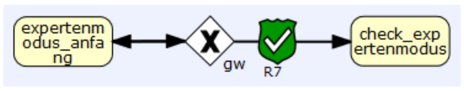
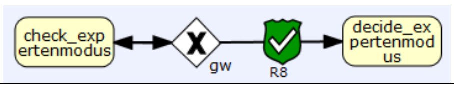
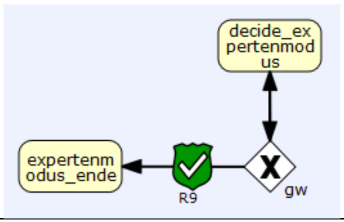
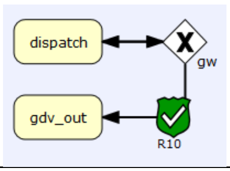
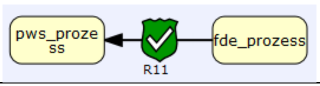
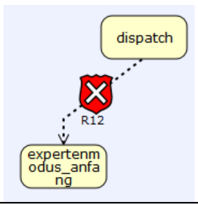
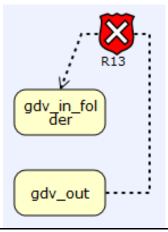


Figure 7.10 The graphical result of rule checking on ARPOS log

Table 7.2 Conclusion of defined rules, descriptions and rule checking on log result

Defined Rule	Description	Rule Checking Result
R1: 	*receive* must precede *check* but *check* does not need to immediately follow *receive*	✓
R2: 	*check* must precede *decide* but *decide* does not need to immediately follow *check*	✓
R3: 	*gdv_in_fol_der* must precede *gdv_gateway* but *gdv_gateway* does not need to immediately follow *gdv_in_fol_der*	✓
R4: 	Bildextraktion_start must precede Bildextraktion_ende but Bildextraktion_ende does not need to immediately follow Bildextraktion_start	✓
R5: 	*decide* must precede *dispatch* but *dispatch* does not need to immediately follow *decide*	✓
R6: 	expertenmodus_anfang must precede expertenmodus_ende but expertenmodus_ende does not need to immediately follow expertenmodus_anfang	✓

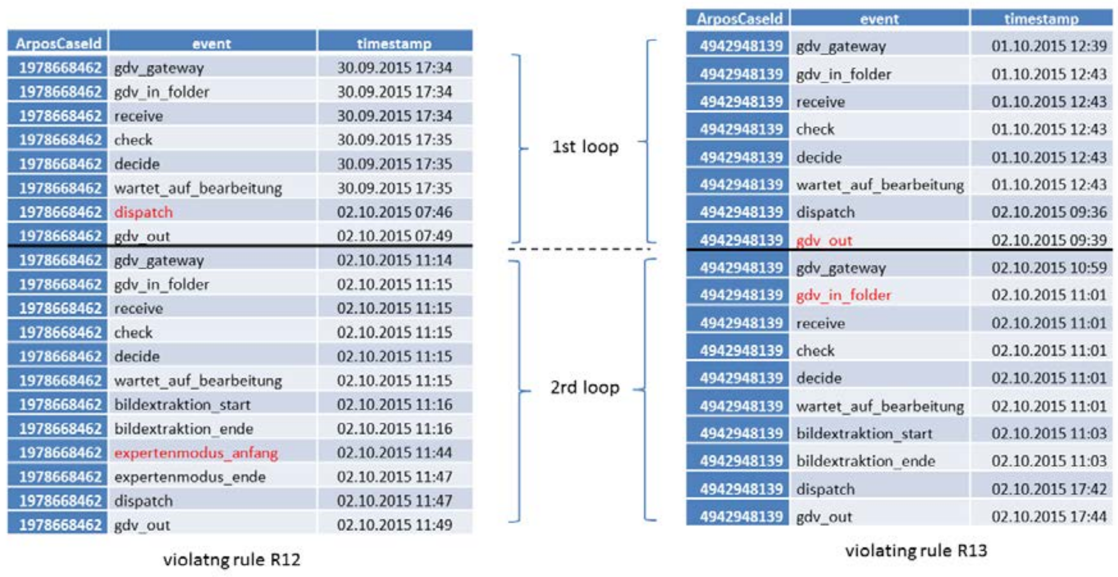
R7: 	expertenmodus_anfang must be immediately followed by check_expertenmodus	✓
R8: 	check_expertenmodus must be immediately followed by decide_expertenmodus	✓
R9: 	decide_expertenmodus must be immediately followed by expertenmodus_end	✓
R10: 	*dispatch* must be immediately followed by *gdv_out*	✓
R11: 	*fde_prozess* must be immediately followed by *pws_prozess*	✓
R12: 	*dispatch* must not be followed by *expertenmodus_anfang* either immediately or later	X
R13: 	*gdv_out* must not be followed by *gdv_in_fol_der* either immediately or later	X

✓ means the compliance checking result is True for all cases i.e. there is no violating behavior occurring in all cases or process instances.

X means the compliance checking result is False i.e. there is some cases do not comply with the rules. Those cases or process instances which do not comply with the rules are shown in the information table result (See section 7.3.2.6 and Figure 7.9).

7.3.2.7 Result Investigation

The developed prototype suggests that there are some violating behaviours detected i.e. some process instances do not comply with the rule R12 and R13. However, after inspecting the violating process instances in ARPOS log, there is not any misbehaviour but they are *loops*. A loop indicates that tasks are repeated i.e. a loop consists of sequence of tasks which are continually executed until satisfying a certain condition.



ArposCaseId	event	timestamp
1978668462	gdv_gateway	30.09.2015 17:34
1978668462	gdv_in_folder	30.09.2015 17:34
1978668462	receive	30.09.2015 17:34
1978668462	check	30.09.2015 17:35
1978668462	decide	30.09.2015 17:35
1978668462	wartet_auf_bearbeitung	30.09.2015 17:35
1978668462	dispatch	02.10.2015 07:46
1978668462	gdv_out	02.10.2015 07:49
1978668462	gdv_gateway	02.10.2015 11:14
1978668462	gdv_in_folder	02.10.2015 11:15
1978668462	receive	02.10.2015 11:15
1978668462	check	02.10.2015 11:15
1978668462	decide	02.10.2015 11:15
1978668462	wartet_auf_bearbeitung	02.10.2015 11:15
1978668462	bildextraktion_start	02.10.2015 11:16
1978668462	bildextraktion_ende	02.10.2015 11:16
1978668462	expertenmodus_anfang	02.10.2015 11:44
1978668462	expertenmodus_ende	02.10.2015 11:47
1978668462	dispatch	02.10.2015 11:47
1978668462	gdv_out	02.10.2015 11:49

violatng rule R12

ArposCaseId	event	timestamp
4942948139	gdv_gateway	01.10.2015 12:39
4942948139	gdv_in_folder	01.10.2015 12:43
4942948139	receive	01.10.2015 12:43
4942948139	check	01.10.2015 12:43
4942948139	decide	01.10.2015 12:43
4942948139	wartet_auf_bearbeitung	01.10.2015 12:43
4942948139	dispatch	02.10.2015 09:36
4942948139	gdv_out	02.10.2015 09:39
4942948139	gdv_gateway	02.10.2015 10:59
4942948139	gdv_in_folder	02.10.2015 11:01
4942948139	receive	02.10.2015 11:01
4942948139	check	02.10.2015 11:01
4942948139	decide	02.10.2015 11:01
4942948139	wartet_auf_bearbeitung	02.10.2015 11:01
4942948139	bildextraktion_start	02.10.2015 11:03
4942948139	bildextraktion_ende	02.10.2015 11:03
4942948139	dispatch	02.10.2015 17:42
4942948139	gdv_out	02.10.2015 17:44

violating rule R13

Figure 7.11 Example of a process instance which violates rule R12 (left) and R13 (right)

The rule R12 expects that *dispatch* must not be followed by *expertenmodus_anfang* either immediately or later. However, the rule is violated, which means that there are some process instances which execute *dispatch* then execute *expertenmodus_anfang* later on. After investigating the log, the cause of violations is loops or re-executions in same process instances shown in Figure 7.11 (left). That is, a particular claim case requires an expert to recheck the case again after the automatic examination on the case done by the system.

Similarly, the rule R13 expects that *gdv_out* must not be followed by *gdv_in_folder* either immediately or later. However, the rule is violated, which means that there are some process instances which execute *gdv_out* then execute *gdv_in_folder* later on. After investigating the log, the cause of violations is loops or re-executions in same process instances shown in Figure 7.11 (right). That is, damaging pictures are received after received a claim. The system receives and then checks a particular case. Later, the system receives damaging pictures so the system has to recheck the case again by investigating also the receiving damaging pictures.

In conclusion, the prototype has a limitation in compliance checking if there are loops occurring in executions. Without this limitation, the result of rule checking on ARPOS log will have no violation and comply with all the defined rules. The ideas to solve the problem can be done with the prototype or ARPOS. The prototype can be added an additional attribute for compliance checking functionalities to indicate that tasks are executed in the second loop. Another possibility is that changing process ids in ARPOS during executing the second loop.

8 Conclusions

In all pervious chapters, this master thesis thoroughly demonstrates related concepts and technologies of the proposed approach of compliance checking for unstructured processes using process mining along with implementation and evaluation with real-life dataset.

In this chapter, the entire research is summarized. Besides, the limitation of the implemented prototype and interesting topics for future developments are pointed out.

8.1 Summary of Contributions

The goal of the master thesis is to leverage the power of process mining in combination with the adaptive case management discipline and develop a prototype solution of the potential combination. The analyzed scenarios are (1) Process mining for supporting knowledge workers in ACM, (2) Process mining for transition from an unstructured to structured process and (3) Process mining for compliance regulations in unstructured processes. The concept of a prototype implementation is based on the three scenarios analysis. By considering the scenarios analysis, a connection between scenarios can be established. That is, the scenario of transition from unstructured to structured process can be the primary step for another two scenarios. Due to the importance of compliance checking in organizations and rare researches on the compliance checking in unstructured processes, therefore this master thesis proposes an approach of *compliance checking for unstructured processes using process mining*.

The prototype implementation consists of two steps. The first step is using process mining to transform an unstructured to structured process in PROM. The used algorithm is heuristic miner due to the ability of discovering main behaviors from an unstructured process. The second step is compliance checking which is implemented in the Oryx platform. It starts with importing a discovered processed model from PROM into Oryx. Due to conflict BPMN version of PROM and Oryx, a converter for transforming PROM BPMN into Oryx BPMN is developed and applied. Graphical rule definition, rule creation, rule checking on log and result visualization functionalities are implemented for compliance checking. The correct and incorrect pattern notations are developed as a new stencil set in Oryx for graphical rule definition. A rule is defined by drag-and-drop a notation on an imported process model. The rule creation translates graphically defined rules into EPL. The rule checking on log validates a given log with the defined rules to detect violations by using Esper CEP engine. The result visualization displays violated rules on the process model and reports all violating process instances combined with the total number of violations for each rule.

The evaluation of the prototype uses a real-life log data extracted from a car claim system of a German insurance company over one-year period. The classical ETL process is employed for log preparation; extracting logs from XML database, transforming XML into CSV with a self-developed converter, and loading to the prototype. The prototype can support a large real-life log data. The evaluation result shows that in 13 defined rules, 11 rules are compliant but 2 rules are violated. The prototype correctly translates graphical rules into the rule language statements. However, the prototype detects the violations due to a limitation of rule checking on a log which has repeated executions. Without this limitation, the result of rule checking on log will have no violation meaning that the execution log of the claiming system complies with all the defined rules.

8.2 Limitations

The prototype is capable with a large real-life log. However, there is a limitation with a log which has repeated executions of the same process instance. With this limitation, the prototype considers the repeated executed tasks as the same iteration as the previous execution. For instance, a rule is defined that A must not be followed by D either immediately or later. The two tasks are executed in different iterations but the same process instance. With this limitation, the prototype concludes that a process instance executes A and later followed by D, so that means the rule is violated even though A and D are actually executed in different iterations. Therefore, the prototype should individually consider each iteration to avoid crossing to other iteration checking.

8.3 Future Work

In this master thesis, the combinations of process mining and ACM focus in the three scenarios: (1) supporting knowledge workers with process mining, (2) transiting from unstructured to structured processes through process mining, and (3) compliance regulations in unstructured processes through process mining. In addition, several possibilities for leveraging process mining in ACM are also pointed out in section 2.3.1. For instance, process mining could be used to generate relationships between resources such as social networks and communication flows among knowledge workers. Also, process mining could discover best practice patterns or templates and share to other knowledge workers.

Continuing from discussing on limitations, the prototype has a starting point for further improvement. That is loop detecting i.e. improving the prototype to be able to use for compliance checking even a log which has repeated executions. The improvement can be done with the prototype or ARPOS. The prototype can be added an additional attribute for compliance checking functionalities to indicate that tasks are executed in the second loop. Another possibility is that changing process ids in ARPOS during executing the second loop.

The rule patterns implemented in the prototype are only subjected to the ordering of tasks. Therefore, more rule patterns could be added for example occurrence of tasks (e.g. Is task P existing or absent?) and time duration aspect (Is task Q executed within one hour?).

Furthermore, the approach of compliance checking for unstructured processes using process mining is integrated with compliance descriptors [55], which is further described in [66].

9 References

[1] R. Agrawal, D. Gunopulos, and F. Leymann, "Mining Process Models from Workflow Logs," *In Sixth International Conference on Extending Database Technology*, vol. 1377, pp. 469–483, 1988.

[2] W. M. P. van der Aalst, *Process mining Discovery, Eonformance and Enhancement of Business Process*. Springer, 2011.

[3] W. M. P. van der Aalst, "Process Mining: Overview and Opportunities," *ACM Transactions on Management Information Systems (TMIS)*, pp. 1–17, 2012.

[4] H. R. Motahari-Nezhad and K. D. Swenson, "Adaptive Case Management: Overview and Research Challenges," *IEEE 15th Conference on Business Informatics (CBI)*, pp. 264–269, 2013.

[5] A. Jalali and I. Bider, "Towards Aspect Oriented Adaptive Case Management," *IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations*, pp. 143–151, 2014.

[6] Workflow Management Coalition, "Comparison: ACM vs BPM," 2010. [Online]. Available: http://www.xpdl.org/nugen/p/adaptive-case-management/public.htm.

[7] N. Palmer, L. Fischer, and S. Reddy, *Thriving on Adaptability: Best Practices for Knowledge Workers*. Future Strategies Incorporated, 2015.

[8] M. Werner and N. Gehrke, "Process Mining," *WISU - die Zeitschrift für den Wirtschaftsstudenten 7/13*, pp. 1–16, 2013.

[9] W. M. P. van der Aalst, "Using Process Mining to Bridge the Gap between BI and BPM," *IEEE Computer*, vol. 44, no. 12, pp. 77–80, 2011.

[10] W. M. . van der Aalst, A. Adriansyah, A. K. . De Medeiros, F. Arcieri, T. Baier, T. Blickle, and B. Chandra, "Process Mining Manifesto," *International*

Conference on Business Process Management. Springer Berlin Heidelberg, 2011.

[11] W. M. P. van der Aalst, M. Pesic, and M. Song, "Beyond Process Mining: From the Past to Present and Future," *International Conference on Advanced Information Systems Engineering.* Springer Berlin Heidelberg, 2010.

[12] H. F. Witschel, T. Q. Nguyen, and K. Hinkelmann, "Learning Business Rules for Adaptive Process Models," 2012.

[13] A. Appice, S. Pravilovic, and D. Malerba, "Process Mining to Forecast the Future of Running Cases," *International Workshop on New Frontiers in Mining Complex Patterns.* Springer International Publishing, pp. 67–81, 2013.

[14] E. Heber, H. Hagen, and M. Schmollinger, "Application of Process Mining for Improving Adaptivity in Case Management Systems," 2015.

[15] S. Huber, M. Fietta, and S. Hof, "Next Step Recommendation and Prediction Based on Process Mining in Adaptive Case Management," *Proceedings of the 7th International Conference on Subject-Oriented Business Process Management.* ACM, 2015.

[16] W. M. P. van der Aalst, A. Weijters, and L. Maruster, "Workflow Mining: Discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering 16*, vol. 9, pp. 1128–1142, 2003.

[17] A. Ehrenfeucht and G. Rozenberg, "Partial (Set) 2-Structures - Part 1 and Part 2," *Acta Informatica 27*, vol. 4, pp. 315–368, 1989.

[18] R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser, "Process Mining Based on Regions of Languages," *International Conference on Business Process Management*, pp. 375–383, 2007.

[19] A. Weijters and W. M. P. van der Aalst, "Rediscovering Workflow Models from Event-Based Data using Little Thumb," *Integrated Computer-Aided Engineering 10*, vol. 2, pp. 151–162, 2003.

[20] C. Günther and W. M. P. van der Aalst, "Fuzzy mining–adaptive process simplification based on multi-perspective metrics," *International Conference on Business Process Management*, pp. 328–343, 2007.

[21] E. Brynjolfsson and A. McAfee, *Race against the machine: How the digital revolution is accelerating innovation, driving productivity, and irreversibly transforming employment and the economy.* Digital Frontier Press, 2011.

[22] M. White, "Case Management: Combining Knowledge With Process," *BPTrends*, pp. 1–14, 2009.

[23] K. D. Swenson, N. Palmer, and B. Silver, *Taming the Unpredictable. Real World Adaptive Case Management: Case Studies and Practical Guidance.* Future Strategies Inc., 2011.

[24] C. Herrmann and M. Kurz, "Adaptive Case Management: Supporting Knowledge Intensive Processes with IT Systems," *International Conference on Subject-Oriented Business Process Management*, pp. 80–97, 2011.

[25] K. D. Swenson, *Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done*. Megan-Kiffer Press, 2010.

[26] M. Kurz, Q. Nrw, W. Schmidt, A. Fleischmann, and M. Lederer, "Leveraging CMMN for ACM Examining the applicability of a new OMG standard for adaptive case management," *Proceedings of the 7th International Conference on Subject-Oriented Business Process Management*, 2015.

[27] "Object Management Group. Case management model and notation (CMMN) Formal Version 1.0," 2014. [Online]. Available: http://www.omg.org/spec/CMMN/1.0. [Accessed: 05-Jul-2016].

[28] M. Hauder, S. Pigat, and F. Matthes, "Research Challenges in Adaptive Case Management: A Literature Review," *EDOC Workshops*, pp. 98–107, 2014.

[29] K. D. Swenson, N. Palmer, and M. J. Pucher, *How Knowledge Workers Get Things Done: Real-world Adaptive Case Management*. Future Strategies Inc., 2012.

[30] A. Oracle, T. Leadership, and W. Paper, "Building the Business Case for BPM," *An Oracle Thought Leadership White Paper*, no. March. 2009.

[31] N. Palmer, "What is BPM." .

[32] Workflow Management Coalition (WfMC), "What is Case Management?," 2010. [Online]. Available: http://adaptivecasemanagement.org/AboutACM.html.

[33] S. Kemsley, "Case management and BPM," *BPTrends*, 2012.

[34] J. Ukelson, "What to do when modeling doesn't work," in *How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done*, Megan-Kiffer Press, 2010.

[35] P. Drucker, *Landmarks of Tomorrow*. Harper & Brothers, New York, 1959.

[36] P. Drucker, *Management challenges for the 21st century*. Harper Collins, 1999.

[37] M. J. Pucher, "Adaptive Process: Theory and Reality," 2010. [Online]. Available: https://isismjpucher.wordpress.com/2010/05/28/adaptive-process-theory-and-reality/.

[38] W. M. P. van der Aalst and C. W. Günther, "Finding structure in unstructured processes: The case for process mining," *Application of Concurrency to System Design*. IEEE, pp. 3–12, 2007.

[39] R. P. Jagadeesh Chandra Bose and W. M. P. van der Aalst, "Abstractions in

process mining: A taxonomy of patterns," *International Conference on Business Process Management*. Springer Berlin Heidelberg, pp. 159–175, 2009.

[40] L. Geng, S. Buffett, B. Hamilton, X. Wang, L. Korba, H. Liu, and Y. Wang, "Discovering structured event logs from unstructured audit trails for workflow mining," *International Symposium on Methodologies for Intelligent Systems*. Springer Berlin Heidelberg, pp. 442–452, 2009.

[41] T. Štajner and D. Mladenić, "Modeling Knowledge Worker Activity," *Proceedings of the workshop on applications of pattern analysis, Cumberland Lodge*, pp. 127–133, 2010.

[42] N. Desai, A. Bhamidipaty, B. Sharma, V. K. Varshneya, M. Vasa, and S. Nagar, "Process trace identification from unstructured execution logs," *IEEE 7th International Conference on Services Computing (SCC 2010)*, pp. 17–24, 2010.

[43] P. M. Esposito, M. A. A. Vaz, S. A. Rodrigues, and J. M. De Souza, "MANA: Identifying and Mining Unstructured Business Processes," *International Conference on Business Process Management*. Springer Berlin Heidelberg, pp. 199–204, 2012.

[44] E. Ramezani, D. Fahland, and W. M. P. Van Der Aalst, "Where did I misbehave? Diagnostic information in compliance checking," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7481 LNCS, pp. 262–278, 2012.

[45] A. Rozinat and W. M. P. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Information Systems*, vol. 33, no. 1, pp. 64–95, 2008.

[46] S. Rinderle-ma, M. Reichert, and B. Weber, "Relaxed Compliance Notions in Adaptive Process Management Systems," 2008.

[47] W. M. P. van der Aalst, B. F. van Dongen, and Günther Christian W., "ProM: The Process Mining Toolkit," *BPM (Demos)*, vol. 489, no. 31. 2009.

[48] A. K. A. de Medeiros, B. F. van Dongen, W. M. P. der Aalst, and A. Weijters, "Process mining: Extending the α-algorithm to mine short loops," *BETA working paper series, WP 113, Eindhoven University of Technology, Eindhoven*, 2004.

[49] A. J. M. M. Weijters, W. M. P. Van Der Aalst, and A. K. A. De Medeiros, "Process Mining with the Heuristics Miner Algorithm," *Technische Universiteit Eindhoven, Tech. Rep. WP 166*, pp. 1–34, 2006.

[50] R. G. Ross, *Principles of the Business Rule Approach*. Addison-Wesley, 2003.

[51] R. Ross, "Business Rules – What You Need to Know," 2016. [Online]. Available: http://masteringbusinessanalysis.com/mba069-business-rules-need-know/.

[52] D. Hay, K. A. Healy, J. Hall, C. Bachman, J. Breal, J. Funk, D. Hay, J. Healy, K. A. Healy, D. Mcbride, R. Mckee, T. Moriarty, L. Nadeau, and S. Quarles,

"Defining Business Rules ~ What Are They Really?," *The Business Rules Group 400*. 2000.

[53] G. Decker, H. Overdick, and M. Weske, "Oryx – An Open Modeling Platform for the BPM Community," *International Conference on Business Process Management*. Springer Berlin Heidelberg, pp. 382–385, 2008.

[54] F. Koetter, A. Weisbecker, and T. Renner, "Business Process Optimization in Cross-Company Service Networks," *2012 Annual SRII Global Conference*. IEEE, pp. 715–724, 2012.

[55] F. Koetter, M. Kochanowski, A. Weisbecker, C. Fehling, and F. Leymann, "Integrating Compliance Requirements across Business and IT," *Enterprise Distributed Object Computing Conference (EDOC), 2014 IEEE 18th International*. IEEE, 2014.

[56] F. Koetter and M. Kochanowski, "A Model-Driven Approach for Event-Based Business Process Monitoring," *Information Systems and e-Business Management*. pp. 5–36, 2015.

[57] S. Sadiq, G. Governatori, and K. Naimiri, "Modeling Control Objectives for Business Process Compliance," *International conference on business process management*. Springer Berlin Heidelberg, 2007.

[58] M. El Kharbili, A. de Medeiros, and W. M. P. van der Aalst, "Business Process Compliance Checking: Current State and Future Challenges.," *MobIS 8*, pp. 107–113, 2008.

[59] F. Koetter, M. Kochanowski, T. Renner, C. Fehling, and F. Leymann, "Unifying Compliance Management in Adaptive Environments through Variability Descriptors (Short Paper)," *2013 IEEE 6th International Conference on Service-Oriented Computing and Applications*. IEEE, 2013.

[60] A. K. Ghose and G. Koliadis, "Auditing business process compliance," *International Conference on Service-Oriented Computing*. Springer Berlin Heidelberg, pp. 169–180, 2007.

[61] K. Namiri and N. Stojanovic, "Towards A Formal Framework for Business Process Compliance," *Multikonferenz Wirtschaftsinformatik*, vol. 259. pp. 1185–1196, 2008.

[62] M. El Kharbili, S. Stein, I. Markovic, and E. Pulvermüller, "Towards a Framework for Semantic Business Process Compliance Management," *Proceedings of GRCIS 2008*. 2008.

[63] C. Giblin, S. Müller, and B. Pfitzmann, "From Regulatory Policies to Event Monitoring Rules: Towards Model-Driven Compliance Automation," *IBM Research Zurich, Report RZ 3662*. 2006.

[64] K. Van Hee, J. Hidders, G. Houben, and J. Paredaens, "On-the-fly Auditing of Business Processes," *Transactions on Petri nets and other models of concurrency*

IV. Springer Berlin Heidelberg, pp. 144–173, 2010.

[65] W. M. P. van der Aalst, H. T. de Beer, and B. F. van Dongen, "Process Mining and Verification of Properties ⊠:*TM* An Approach based o *Confederated International Conferences" On the Move to Meaningful Internet Systems*. Springer, 2005.

[66] F. Koetter, M. Kintz, M. Kochanowski, C. Fehling, P. Gildein, S. Wagner, T. Wiriyarattanakul, F. Leymann, and A. Weisbecker, "An Universal Approach for Compliance Management using Compliance Descriptors." pp. 1–22.

[67] "Pentaho Data Integration (Kettle) Tutorial." [Online]. Available: http://wiki.pentaho.com/display/EAI/Pentaho+Data+Integration+(Kettle)+Tutorial.

[68] "EsperTech Event Series Intelligence." [Online]. Available: http://www.espertech.com/.

[69] T. Renner and J. Finzen, "„Automatische regelbasierte Prozess- steuerung zur Onlineabwicklung von Schadensfällen"." 2011.

[70] T. Davenport, *Thinking for a living: how to get better performances and results from knowledge workers*. Harvard Business Press, 2005.

[71] T. Davenport and M. Beers, "Improving Knowledge Work Processes By," no. November, 1996.

[72] T. H. Davenport and L. Prusak, *Working Knowledge: Managing What Your Organization Knows*. 1998.

[73] J. Ukelson, "Unstructured, Semi-Structured and Structured Processes," 2009. [Online]. Available: http://exeedtechnology.com/unstructured-semi-structured-and-structured-processes.

[74] M. Rath and R. Sponholz, *IT-Compliance: Erfolgreiches Management regulatorischer Anforderungen*. Erich Schmidt, 2009.

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Stuttgart.

Thatchanok Wiriyarattanakul

Stuttgart, 7 October 2016