Artifact-centric log extraction for cloud systems

Santana Calvo, H.A.

Award date:
2017

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Architecture of Information Systems Research Group

Artifact-Centric Log Extraction for Cloud Systems

Master Thesis

Hugo Andres Santana Calvo

Supervisors:

Dr. ir. Dirk Fahland
Dr. ir. Alexander Serebrenik
Dr. ir. George Fletcher

Final Version (Public)

Eindhoven, August 2017

Abstract

Process Mining is a fast-growing discipline that has the end goal of improving business processes for enterprises. It can be described as a three phase procedure consisting of extraction of information, transformation of information and producing findings and insights using different process mining techniques. During this project we produce a complete process mining analysis of cloud systems by covering these three phases. Moreover, special attention is given to the transformation of information as traditional methods are time-consuming and require a lot of technical expertise such as SQL scripting knowledge. A prototype based on the *"Artifact-Centric Approach"* and the work of E.H.J. Nooijen et al. [34] [35] and Xixi Lu et al. [26] [27] was developed. New features were included to manipulate the data and artifacts in order to solve the *"Vertical Anti-Partitioning"* challenge and accept domain knowledge from the users. By using this prototype, we are able to obtain the *same results* as traditional approaches in the transformation step while also automating the procedure and reducing the required SQL knowledge. Additionally, an API extractor and different process mining analyses were created to produce the end to end cycle of process mining for cloud systems. The technique was validated with two cloud systems. All the obtained results and software tools were developed by me as both closed-source and open-source projects.

Keywords: Process Mining, Artifact-Centric Approach, Cloud Systems

Acknowledgements

First, I would like to thank my mom, my dad and my brother who have been supporting me during all these years. Without you I would not be here today.

Secondly, I would like to thank my supervisor, Dirk Fahland. None of this would have been possible without your weekly guidance, patience and open mindedness along the entire process.

To all my friends for listening to my crazy ideas, stories and contributing one way or another, thanks a lot! Estefania, Ishaan, Alberto, Juan, Angelo, Tia Lucia, Sra Palerma and Oriana.

Last but not least, I would like to thank The European Institute Of Technology (EIT Digital) for creating this amazing program that allowed me to study in two great Universities (Technical University of Eindhoven and Technical University of Madrid), gave me the chance to experience many cultures and live in three different countries. I am very proud to be part of the EIT family.

Contents

Contents vii

List of Figures xi

List of Tables xiii

Listings xv

1 Introduction 1

1.1 Motivation . . . 1

1.2 Research Problem and Research Questions . . . 2

1.3 General Approach and Validation Methodology . . . 3

1.4 Thesis Outline . . . 4

2 Background and Previous Work 7

2.1 What is Process Mining? . . . 7

2.2 Traditional Approach . . . 8

2.3 The Artifact-Centric Approach . . . 9

2.4 Other Approaches . . . 11

2.4.1 Xesame . . . 11

2.4.2 OpenSlex . . . 12

2.4.3 Ontology Based . . . 12

2.5 Process Mining for Cloud Systems . . . 12

3 Requirements 15

4 End to End Process Mining for Cloud Systems 17

4.1 Process Mining Phases for cloud systems . . . 17

4.1.1 Extraction Phase . . . 17

4.1.2 Transformation Phase . . . 18
4.1.3 Analysis Phase . . . 19
4.2 Technical Setup . . . 19
4.2.1 Cloud Systems . . . 19
4.2.2 API Extractor . . . 20
4.2.3 SQL Scripts . . . 20
4.2.4 XTract . . . 20
4.2.5 Process Mining Analysis . . . 21
4.2.6 Databases . . . 21
4.3 Modification Stages . . . 22
4.4 Needed Features . . . 24
4.4.1 Extraction Phase . . . 25
4.4.2 Transformation Phase . . . 25
4.4.3 Analysis Phase . . . 28

5 Extractor Design and Implementation **29**
5.1 API Extractor Operation . . . 29
5.2 Refinement of needed features in the Extraction Phase . . . 33

6 Event Log Extraction Using the Artifact-Centric Approach **35**
6.1 XTract V1 Evaluation . . . 35
6.1.1 Features XTract V1 is able to accomplish . . . 35
6.1.2 Features XTract V1 is not able to accomplish . . . 36
6.2 Artifact-Centric Approach Additions . . . 38
6.2.1 Data Preparation . . . 38
6.2.2 Manual Database Schema Identification . . . 42
6.2.3 Artifact Manipulation . . . 42
6.2.4 Export Activities Table . . . 44

7 Transformation Phase Evaluation **49**
7.1 Cloud Systems Structure and Extracted Information . . . 49
7.1.1 Jira . . . 49
7.1.2 Salesforce . . . 50
7.2 Traditional Approach . . . 52
7.3 Artifact-Centric Approach with XTract 3 . . . 52

7.4 Evaluation of Results 59
7.4.1 First Level Results - Basic Event Log 59
7.4.2 Second Level Results - Complete Event Log 59
7.5 How much effort is needed? 60

8 Process Mining Analysis **63**
8.1 Refinement of needed features in the Analysis Phase 63
8.2 Analysis 64
8.3 Data Sets Constraints 64
8.4 Findings 65

9 Process Dimensions **67**
9.1 Process Dimensions 67
9.2 Changing Dimensions with XTract V3 68
9.3 Advantages 68

10 Conclusion **69**
10.1 Summary of Contributions 69
10.2 Future Work 70

Bibliography **71**

List of Figures

1.1 Bottleneck in the Transformation Phase. 2
1.2 Technical Setup. 4
1.3 Thesis topics and chapters. 5

2.1 Event log example. 8
2.2 Artifact-Centric Approach original method. 10
2.3 Interacting artifacts visualization. 11

3.1 Sorting Attribute example. 16

4.1 Process Mining Phases. 18
4.2 Technical Setup. 19
4.3 Extraction Database and Transformation Database. 21
4.4 Change Table representation options. 26
4.5 Repeated Activities in Cloud Systems. 27
4.6 Activities granularity example. 28

5.1 API extractor overview. 30

6.1 Multiple objects represented in the same table example. 37
6.2 Artifact-Centric Approach original method. 38
6.3 Interfaces to remove tables or columns. 39
6.4 Granularity selection. 41
6.5 Manual primary keys selection. 42
6.6 Manual foreign keys selection. 42
6.7 Main table and secondary tables selection. 43
6.8 Remove activities interface. 44
6.9 Activities table name and Extraction Database credentials. 45

6.10 Artifact-Centric Approach New Steps. 47
6.11 XTract V3 Additions. 48

7.1 Jira extracted relational database, only relevant columns are shown. 50
7.2 Salesforce extracted relational database, only relevant columns are shown. 52
7.3 Selection of tables to split and their primary keys. 53
7.4 Selection of field, from and to columns. 53
7.5 Granularity level selection. 54
7.6 Manual selection of primary key. 55
7.7 Foreign Keys Jira. 56
7.8 Artifact discovery results. 56
7.9 Adding Sorting column for each activity. 57
7.10 Editing event names. 57
7.11 Selecting additional attributes. 58
7.12 Traditional Approach Results. 60
7.13 Artifact-Centric Approach Results. 61

8.1 Open Issues key performance indicators. 64
8.2 Resolution Time key performance indicators. 64
8.3 Delays caused by change of priority or change of assignee. 65

9.1 Case Id Selection. 67
9.2 Graphical interface to edit an artifact. 68

List of Tables

3.1 High-level requirements summary. 16

4.1 Jira Application's Family 20
4.2 Salesforce Application's Family 20
4.3 Modification Stages, data sources and process mining phases. 23
4.4 Needed features in the Extraction Phase. 24
4.5 Needed features in the Transformation Phase. 26
4.6 Needed features in the Analysis Phase. 28

6.1 Features XTract V1 is able to accomplish 36
6.2 Features XTract V1 is *not* able to accomplish 36
6.3 Proposed Granularity Levels. 41
6.4 Example of naming convention after a split. 41
6.5 Summary of new steps in the Artifact-Centric Approach 46
6.6 XTract V3 Additions from figure 6.11 48

7.1 Amount of rows and columns per table for Jira. 51
7.2 Amount of rows and columns per table for Salesforce. 51
7.3 Obtained result with the Traditional Approach (Simple Event Log). 59
7.4 Obtained result with the Artifact Centric Approach (Simple Event Log). 60
7.5 Non-automatic actions summary. 61

8.1 Developed analyses. 63

Listings

5.1 API response. 30
5.2 Java Object example. 31
5.3 Processing objects. 32
5.4 Tables selection. 32
6.1 Joining changes tables pseudo code (Representation Option 2). 39
6.2 Split tables and granularity levels definition pseudo code. 40
6.3 Add static attributes pseudo code. 44
6.4 Basic event log generation pseudo code. 45
6.5 Pseudo code to add attributes in the activities table. 46

Chapter 1

Introduction

The present master thesis was completed as part of the Data Science Master at Eindhoven University of Technology (TU/e) and realized with the Architecture of Information Systems (AIS) group of the Mathematics and Computer Science department of TU/e and an IT company.

This chapter aims to introduce the master thesis. Section 1.1 presents a discussion about the motivation to complete the project. From the academia, important publications and known challenges in the process mining field were used as inspiration. Real world requirements were gathered at the IT company. The research problem and research questions will be illustrated in Section 1.2. In Section 1.3 the selected approach to solve the research problem and the validation methodology will be presented. Finally, Section 1.4 displays an outline of how the rest of the chapters are structured. All the obtained results and software tools were developed by me as both closed-source and open-source projects.

1.1 Motivation

Process Mining is a fast-growing discipline that can be considered the linking bridge between business processes and business intelligence. The end goal of this discipline is to improve business processes for enterprises. In order to reach this goal, one must follow several required steps that have distinct levels of complexity. Data storage, data transmission, data transformation and data analysis are all equally important when it comes to improve business processes. However, much of the research efforts have been spent on data analysis, without much effort being applied in the data transformation. In this field, data transformation refers to the procedure of converting raw data stored in information systems into a well-defined format called an *event log*. Wil van der Aalst acknowledged the key role of event logs in process mining by stating "process mining stands or falls with the availability of event logs" [45]. Without event logs process mining cannot be duly performed, since they serve as the starting point of process mining analysis.

The generation of event logs is not a trivial task, as their quality depends highly on the underlying structures of the systems holding the information. The complexity is stated in The Process Mining Manifesto. This document is a public declaration of principles and intentions developed by the members and supporters of the IEEE Task Force on Process Mining [46]. It defines five Maturity Levels of event logs. These levels of maturity aid greatly in understanding the complexity behind the creation of event logs. The lowest level of maturity refers to a scenario where the information is stored by hand in a non - systematic way. The highest level of maturity is related to information systems where all the actions and attributes are automatically stored with clear semantics. Since it is not possible to predict if the highest level of maturity will ever

be reached at a high scale, more effort needs to be applied into the transformation step. This can be achieved by improving current transformation techniques. The whole approach for event log generation needs to be evaluated, not only the final output. Elements such as information systems design, domain knowledge, computation time and automation need to be taken into account to produce an efficient solution. Traditionally, event logs are extracted through manual construction of SQL queries [25]. A visual representation of the process mining steps can be seen in figure 1.1.

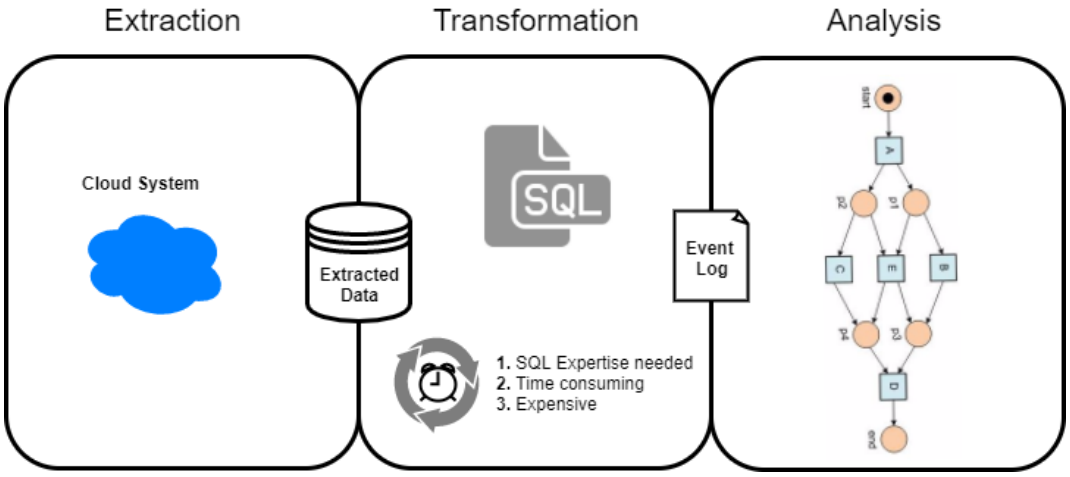

Figure 1.1: Bottleneck in the Transformation Phase.

There are many sources explaining how to obtain event logs from ERP systems [37], [39], [25] and [19]. These examples are affected by different challenges. The approaches followed by [37],[25], [19] could obtain event logs with the trade-offs of requiring plenty of manual effort, SQL scripting knowledge or domain knowledge. All these techniques can be categorized as traditional approaches. Erik H. J. Nooijen proposed a technique called the "Artifact-Centric Approach" [34], which can to automate many steps in the process. Furthermore, it can handle the SQL scripting at an internal level with less user interaction than the traditional approaches. Nevertheless, this technique has been catalogued as complicated by the end user and rarely used [45]. In the present work, a prototype was built to demonstrate how this technique can be used as the backbone for the generation of event logs. The prototype is able to remove some of the constraints of the traditional approaches, such as SQL expertise and time spent by automating different parts of the procedure. The prototype is named XTract v3 and is based on the work of Erik H. J. Nooijen (XTract v1) [34] and Xixi Lu (XTract v2) [26].

As previous research has focused exclusively on on-premise systems such as SAP and Oracle, this project takes a different route. Cloud systems are increasing their usage at a fast pace as more companies are opting to move to the "Digital World". As it was mentioned by van der Aalst [42], cloud computing and Software as a Service (SaaS) take advantage of the possibility to share infrastructure and software across multiple organizations. The end result is a reduction in setup times, pay per use reducing prices and less maintenance and management efforts. In spite of the current momentum of cloud systems and the different advantages they provide, there is not much process mining research in the area. For this reason, two popular cloud systems (Salesforce and Jira) were selected as the target source systems for the current study.

1.2 Research Problem and Research Questions

In Section 1.1 the motivation for completing the project was presented. After taking into account process mining challenges in the transformation phase and requirements gathered at the IT

company, we define the research problem as follows:

> *Given a **cloud system**, improve the transformation step of the raw data into event logs by providing a methodology, which can A) reduce the expertise required to transform the data, B) produce the same results as the current traditional approach used by the engineers C) be flexible enough to allow for domain knowledge decisions.*

As it was mentioned in Section 1.1, the Artifact-Centric Approach has been selected to solve the problem. The reasons for this decision will be provided in Chapter 2. With this in mind, the stated research problem can be divided into the following research questions:

1. Can the Artifact-Centric Approach be used to generate event logs that are equal as the ones generated by the Traditional Approach?
2. What level of automation can be obtained with the Artifact-Centric Approach?
3. To what extend can the Artifact-Centric Approach give new perspectives on the data?
4. What kind of insights can be gained on data from cloud systems?

The first question is very important, because if the proposed technique is not able to reproduce the expected results then it cannot be used. The next two questions look into additional advantages from using the Artifact-Centric Approach. If the first question is satisfied, then answering questions 2 and 3 could provide additional benefits. Finally, the last question looks to understand if process mining for cloud systems provide insights and relevant results.

1.3 General Approach and Validation Methodology

The aim of the following section is to give an overview of the selected procedure to produce an end to end analysis of cloud systems. End to end refers to every step involved to obtain insights and findings using process mining techniques. The first step consists of generating general requirements. After the general requirements are defined, an analysis of existing tooling was made resulting in the selection of the Artifact-Centric Approach as the core technology for generating event logs. The proposed technical setup is shown in figure 1.2. The next step is the conversion of the general requirements into very specific features. These features will be used to understand what is already developed and what needs to be integrated into XTract v3 or in any other component of the technical setup. Once the specific requirements are completed, a step by step guide across the three process mining phases for Jira and Salesforce is shown.

All results of the proposed procedure are finally implemented in supporting software. All steps of the transformation phase are implemented as contribution to the open-source tool XTract, leading to version 3.0, source-code available at [1]. All steps of the extraction and analysis phase are implemented as closed-source tools in the IT company.

[1] https://svn.win.tue.nl/repos/prom/XTract/

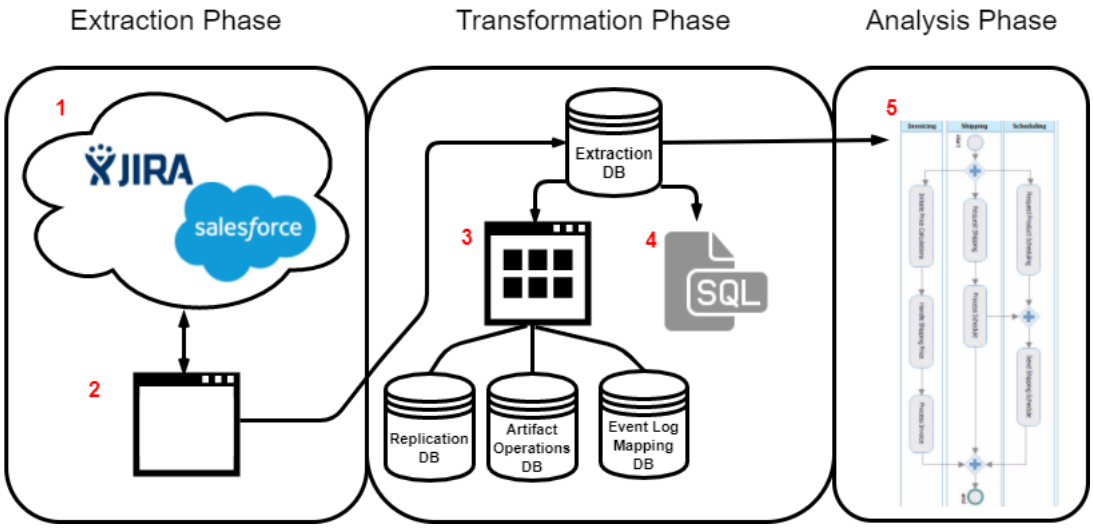

Figure 1.2: Technical Setup.
1: Cloud Systems 2: API Extractor 3: XTract v3
4: SQL Scripts 5: Process Mining Techniques

1.4 Thesis Outline

The remainder of this report is structured as follows. In Chapter 2, detailed explanation about key process mining concepts and relevant background information will be presented. Topics such as the challenge of transforming information into an event log, the original Artifact-Centric Approach, the traditional transformation and relevant cloud system technologies will be presented. Chapter 3 displays all the general requirements. Chapter 4 provides the proposed refinement of process mining for cloud systems. The differences between the process mining phases, definitions of each component in the technical setup, and the conversion of general requirements into specific features is discussed.

Chapter 5 and 6 consist of the developed technical contributions. In Chapter 5 the design and implementation of the API extractor for the Extraction Phase is shown. Chapter 6 introduces the necessary additions to the Artifact-Centric Approach to complete the needed features. Although the general core of the technique remains the same, new features were needed. Chapter 7 displays the case studies. This chapter focuses on the Extraction Phase and the Transformation Phase. Furthermore, the results are shown and an analysis on how much manual effort is required when using the Artifact-Centric Approach is presented

Chapter 8 covers the process mining analysis to demonstrate the value of process mining. Besides the process mining analysis, Chapter 9 provides additional benefits of the Artifact-Centric Approach in the form of dimension changes. A brief description of the concept and the new available insights are made clear in this chapter. Finally, a summary of contributions and future work is exposed in Chapter 10.

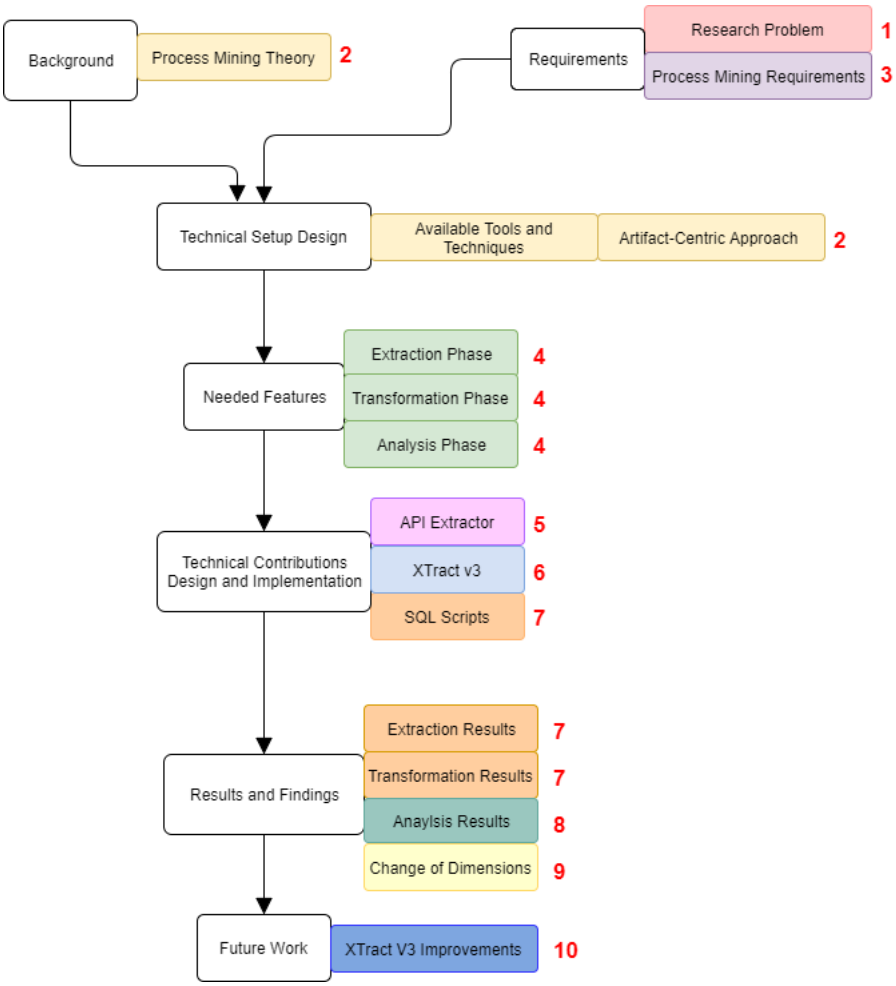

Figure 1.3: Thesis topics and chapters.

Chapter 2

Background and Previous Work

The following chapter will present the most relevant techniques and concepts needed to understand the rest of the thesis. Section 2.1 describes process mining by introducing the most important elements in this discipline with special attention to the generation of event logs. After stating the most important process mining definitions, the traditional approach to obtain event logs will be summarized. Section 2.3 produces a description of the Artifact-Centric Approach. A discussion about each step of the approach, the known issues, and reasoning on why it will be the core technology for the project will be shown. Furthermore, a description of other techniques and why they were not used will be shown. Section 2.5 provides relevant concepts about cloud computing such as application programming interface (API). Additionally, related work found in the literature about process mining for cloud systems will be shown.

2.1 What is Process Mining?

Process mining can be defined as a bridge that connects traditional model-based process analysis (e.g. simulation and other business process management techniques) and data-centric analysis techniques such as machine learning and data mining [43]. The general idea behind the technology consists of detecting and extracting *digital footprints* stored in information systems. Digital footprints are the actions that were executed in the given system. The availability of these actions is tied to the sophistication of the information system. However, it is potentially feasible to track activities such as customers purchasing items on a website, automatic teller machines accepting cash withdrawals or editing the price of a purchase order transaction in a CRM system. Once this information is extracted, cleaned and transformed into a format called event logs, a model describing the behavior of the process can be discovered. There is plenty of research in the topic, with a high amount of publications about process discovery [43], [48], [50], [5], [21], [6], [11], [12] [15], [22], [17]. The real advantage of process mining comes after a model is discovered. Techniques such as conformance checking [47], [4], [3], [2], [16], [31], [52] and social interactions [49], [41], [24] provide powerful insights on what is occurring in business processes. Additionally, throughput times between activities can be easily seen, finding bottlenecks and process irregularities. All these techniques are empowered by the possibility to analyze high amount of data at once, making them much more effective than old process improvements methods based on user interviews or virtual processes that were not created with real data.

As information systems evolve and become more robust, the amount of digital footprints increases. This improves the data sets and opens doors regarding process mining analysis, but also increases the challenge of data extraction, data cleaning and data transformation. The creation of event logs is a complicated task, which has been discussed in multiple studies [26], [34],

[27],[35],[9],[46],[37],[39],[25],[10],[45],[19]. Figure 2.1 presents a fragment of an event log.

_CASE_KEY	ACTIVITY_EN	EVENTTIME
50024000001A7SAAA0	Create Case	2015-06-07 18:16:00.000
50024000001A7SAAA0	Change Status to Closed	2015-06-07 21:35:50.000
50024000001A7SAAA0	Change Status to Approved	2015-06-07 21:50:31.000
50024000001A7SAAA0	Change Status to Closed	2015-06-17 13:09:24.000
50024000001A7SjAAK	Create Case	2015-06-07 18:20:47.000
50024000001A7SjAAK	Change Status to Closed	2015-06-07 21:35:42.000
50024000001A7SjAAK	Change Status to Approved	2015-06-17 13:05:25.000
50024000001A7SjAAK	Change Status to Closed	2015-06-17 13:09:30.000
50024000001A7TIAA0	Create Case	2015-06-07 18:25:15.000
50024000001A7TIAA0	Change Description	2015-06-07 18:30:15.000
50024000001A7TIAA0	Change Status to Closed	2015-06-07 21:35:34.000

Figure 2.1: Event log example.

The following definitions are obtained from [43] and are vital to understand the objective of the current work.

Definition 2.1.1 *(Simple event log)*

Let $\mathscr{A}$ be a set of activity names. A simple trace o is a sequence of activities, i.e., $o \in \mathscr{A}^*$. A simple event log L is a multi-set of traces over $\mathscr{A}$, i.e., $L \in \mathbb{B}(\mathscr{A}^*)$. [1]

The same concept applies for activities tables. Activities tables are event logs stored in relational databases.

Definition 2.1.2 *(Event, attribute)*

Let $\mathscr{E}$ be the event universe, i.e., the set of all possible event identifiers. Events may be characterized by various attributes, e.g., an event may have a timestamp, correspond to an activity, is executed by a particular person, has associated costs, etc. Let AN be a set of attribute names. For any event $e \in \mathscr{E}$ and name $n \in AN$, $\#_n$ (e) is the value of attribute n for event e. If event e does not have an attribute named n, then $\#_n(e) = \perp$ (null value).

Definition 2.1.3 *(Case, trace) Let $\mathscr{C}$ be the case universe, i.e., the set of all possible case identifiers. Cases, like events, have attributes. For any case $c \in \mathscr{C}$ and name $n \in AN$: $\#_n$ (c) is the value of attribute n for case c ($\#_n(c) = \perp$ if case c has no attribute named n). Each case has a special mandatory attribute trace, $\#_{trace}(c) \in \mathscr{E}^*$.* [2] *$\hat{c} = \#_{trace}(c)$ is a shorthand for referring to the trace of a case.*

A trace is a finite sequence of events $o \in \mathscr{E}^*$ such that each event appears only once, i.e., for $1 \leq i < j \leq |o| : o(i) \neq (j)$.

2.2 Traditional Approach

The traditional approach can be described as the classical method to obtain event logs. The closest publication explaining a framework for event log extraction was created by Mieke Julie Jans [25].

Usually, the information is stored in a relational database because most of the information systems use this type of database. Consequently, a SQL scripting approach is used. The first step consists of understanding the relational database structure. Relevant tables and the interaction in the form of foreign key relations should be identified. The next step is the selection of the case id.

[1] Note that we still assume that each trace contains at least one element, i.e., o ∈ L implies o ≠ ⟨⟩

[2] In the remainder, we assume $\#_{trace}(c) \neq \langle\rangle$, i.e, traces in a log contain at least one event.

The case id or process instance is the primary key of one of the tables in the system. This is a key decision since it will dictate the entity or object to be analyzed. Depending on the process, the case id selection could be a combination of tables. Furthermore, there are processes were the case id is very intuitive and not a difficult selection. The last step of the method before writing SQL scripts is the identification of the activities. The activities are identified by time stamps. Usually, there is a changes log table, which contains all the changes in the system. More details about this type of table is presented in Section 4.1.1.

In order to detect the activities, a query to join the case id table to the table that holds the activity information is required. Depending on the complexity of the system, these operations could be complicated. The result of these queries usually is made of the activity name and the time stamp of the action. However, there are cases where additional attributes are added to the event log. In those cases, the queries need to specifically select the additional attributes.

The described procedure needs a lot of effort because these queries need to be written manually. Furthermore, domain knowledge is required to understand which activities and attributes are relevant for the specific process. The final event log is a product of many iterations involving different stakeholders such as SQL experts (data scientists or business analysts), database administrators and the business owner. Real examples from the case studies are provided in the step by step guide in Section 7.2.

2.3 The Artifact-Centric Approach

During this section, we discuss the selected technique for solving the generation of event logs. We start the section by taking a look at the origins of the procedure, before going through each of the steps. The multiple challenges affecting the methodology will be exposed in combination with recent work in the area. The reasons for using the Artifact-Centric Approach will be stated at the end of the section.

As it can be inferred by its name, this approach is using artifacts. The definition of artifacts was provided by Nigam et al [33] from IBM. *An artifact is a concrete, identifiable, self-describing chunk of information that can be used by a business person to actually run a business.* Moreover, a second definition was provided by Cohn et al [14] *Artifacts combine both data aspects and process aspects into a holistic unit, and serve as the basic building blocks from which models of business operations and processes are constructed.* The concept of an artifact goes hand in hand with the artifact life-cycle. *Artifact processing is a way to describe the operations of a business. An artifact life cycle captures the end-to-end processing of a specific artifact, from creation to completion and archiving*[33].

Erik H. J. Nooijen created an automatic procedure which combined these concepts with multiple ideas from different fields. The objective of his procedure was to extract event logs from relational databases. He developed a prototype called XTract, which we address as XTract v1 [34], [35]. In addition to the artifacts and artifact life-cycle discovery, he designed and implemented several steps to enable the generation of automatic event logs. Figure 6.2 displays a graphical representation of all the steps.

Schema Extraction

Schema extraction is defined here as extracting structural information (e.g. candidate keys and relationships between entities) from structured data (e.g. tables). It takes as input a set of structured (tabular) data and produces (1) a primary keys for each table, (2) the domain for each element (column) in each table and (3) the relationships (foreign keys) between tables.

Identify artifact schemas

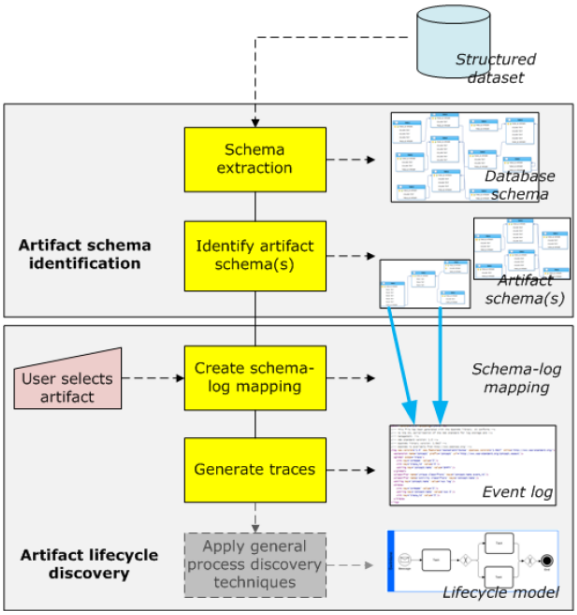

Figure 2.2: Artifact-Centric Approach original method.

In this step the subset of the dataset that is of interest for each specific artifact is identified. The idea is to partition the full schema in a number of clusters (one for each artifact) and assign each table to one or more clusters. In addition, a representative main table will be chosen for each cluster. This main table contains the instance information for the specific artifact: each artifact instance can be identified by the primary key of the main table.

Create schema-to-log mapping

Creation of the mapping between the artifact schema and a resulting event log is done in this step. It takes as input (1) an artifact schema and (2) a structured dataset described by the artifact schema and produces (1) a set of event types identified in the dataset and (2) a mapping from the dataset to these event types. The mapping found in this way describes how to extract the different events from the dataset. Note that in this case the purpose of the event log is to use it to discover the lifecycle of an artifact using process mining techniques. Since it depends on the goal of a process mining project what to include in an event log [46], [9] this should be taken into account.

Generating traces

Using the mapping found in the previous step events can be generated from the dataset. The approach described in [9] was designed for a similar purpose: it takes as input a dataset and a mapping and produces an event log. Similarly for each artifact this step should take as input the dataset and the mapping for the artifact as described in the previous section. The output should be an event log in the eXtensible Event Stream (XES) format; The XES format was chosen since this is the only standardized event log format. It was designed for the interchange of event log data in a simple, expressive and flexible format, while allowing for extensions of the format in a transparent manner [13]. An approach to generating traces given a mapping and a dataset is described in Section 4.3.

For our technical setup, we are not interested in generating the output in XES format. A discussion about this point is presented in the general requirements from Chapter 3.

Apply process discovery techniques

The goal of artifact lifecycle discovery is to discover both the internal lifecycle of an artifact

and its interaction with other artifacts, thereby fully describing how an artifact operates. With the event log produced in the previous step a variety of process discovery techniques can be used to generate the lifecycle for each artifact.

Xixi Lu announced several issues about XTract v1 [26]. These issues can be summarized as 1) Unable to identify complex foreign keys, 2) Unable to identify multiple artifacts within one table and multiple event types within one column, 3) Performance issues related to log extraction and 4) No interactions between artifacts. These issues are addressed by creating interactions between artifacts. Since each artifact produces an event log, an artifact can be considered a process. Consequently, artifact interactions aim to discover very specific artifacts and then find the relation within the activities. Although this new procedure is able to address most of the problems, it produces two challenges. The most notorious one is that there is not a clear way to visualize these artifacts and their interactions. Secondly, defining interactions between artifacts need to be completed manually by the end user, resulting in a complicated and non-automatic method. Figure 2.3 displays a graphical representation of multiple interacting artifacts from a high-level view. In addition to these challenges, the prototype created by Xixi Lu (XTract v2) is private and the source code is not available.

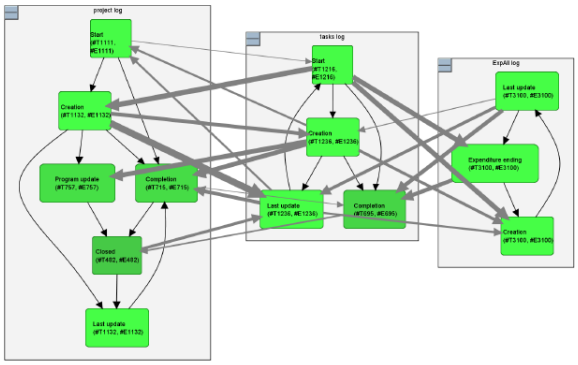

Figure 2.3: Interacting artifacts visualization.

Even though the exact work from [26] cannot be re used, many of the findings and XTract v1 analysis can be used as a guideline to solve the stated known issues. As the main goals of the current project go hand in hand with saving time, reducing amount of complexity and producing correct event logs in the Transformation Phase, the Artifact-Centric Approach seems to be an effective technology to archive those objectives.

2.4 Other Approaches

The current section dives into different technologies that were studied before the selection. The reasons for discarding them will be stated.

2.4.1 Xesame

Xesame (Previously called XESma or XES Mapper) was created by J.C.A.M. Buijs [9]. The goal of this prototype was to create a software program to allow the mapping of information systems databases into XES files. Some parts are used in XTract v1. XTract adds additional capabilities such as the schema-to-log-mapping, which improves automation. Although Xesame is not used directly, the concepts and observations developed in the mentioned work are indirectly employed.

2.4.2 OpenSlex

OpenSlex is a meta model introduced by Murillas et al [18]. This meta model provides a universal technique that can be used to separate data analysis and data extraction. The concept of the data model on top of information systems allows new features such as an easy way to change perspectives on an event log. However, there is a very fundamental limitation, in order to convert the data to introduce it into the OpenSlex data model, data transformation from the different systems needs to be done. Consequently, this technique does not relate to the objectives of the current project.

2.4.3 Ontology Based

An ontology-based procedure was proposed by A. Syamsiyah to extract event logs from relational database systems [10]. Although the method proved to generate event logs in the form of XES files, there are several limitations. On one hand, event related notions need to be manually defined by experts in the domain ontology. On the other hand, users require SPARQL expertise to write the annotations in the system. Therefore, this technique is discarded.

2.5 Process Mining for Cloud Systems

The National Institute of Standards and Technology (NIST) defines cloud computing as *a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction* [29]. The possibilities that cloud computing brings in terms of efficiency and accessibility has created an incredibly amount of services such as Amazon EC2/S3, Google Apps, Microsoft Azure. These services provide the needed infrastructure for systems with different specializations such as CRM (SalesForce, SAP Ariba, SugarCRM, Microsoft Dynamics CRM), storing services (Dropbox, Box), human resources (SAP Success Factors, Workday, Cornerstone) among many other fields. Wil van der Aalst mentioned the advantages of cloud computing and configurable services in the cloud. He proposed the term *"cross-organization process mining"* to take advantage of cloud computing and cloud systems to allow organizations to learn from each other and improve their processes [42].

In the mentioned publication, the work of Dustdar et al was mentioned. In his research, Dustdar et al applied data mining and process mining techniques to web services [20]. They proposed a method called Web Services Interaction Mining (WSIM). However, this mechanism cannot be used for our goals since there is not a defined framework that can be easily adapted in the form of a centralized tool. The approach extracts information from heterogeneous sources and requires extensive development efforts. Web scrapping encounters similar problems.

When looking at standard ways to extract the information stored in cloud systems the most common procedure uses Application Programming Interfaces (API). These interfaces provide a mechanism for code reuse so programmers can build on top of the work that other programmers (or they themselves) have already done, rather than starting from scratch with every program. Furthermore, using APIs is often required, because low-level access to system resources (such as graphics, networking, the file system, etc.) is only available through protected APIs [32]. Nevertheless, APIs highly depend on their level of sophistication. Each system owner is responsible for APIs documentation and maintenance over time. Robillard et al stated different reasons why these interfaces can be hard to learn and documentation can be a huge factor in decreasing programmers' productivity [40].

From the literature review we can conclude that application programming interfaces are the

best option to extract the data from cloud systems and an *API Extractor* needs to be developed for each individual system. However, developing this extractor could be challenging because of factors such as documentation and non-public access to key data. To understand this challenge, it is necessary to compare the extraction differences between cloud systems and on-premise systems. When looking at the most popular on premise systems (i.e. SAP and Oracle), there are tools which handle the extraction of information by replicating the relational databases. It is possible to filter the extracted data by specifying parameters such as the date, but the underlying relational database structure will remain the same. This was the case in [26]. Specific details of the API extractor for the case studies is presented in Chapter 5.

The second challenge is the transformation of the extracted information into a suitable event log for process mining analysis. A complete overview of this challenge for ERP systems have been provided in [27]. In Section 4.1, named *"Relational Schemas vs. High-Level Models"* Xixi Lu et al. provide a complete overview of the needed operations to transform relational schemas into high-level models. High-level models can be defined as efficient data representation structure for process mining purposes. The following data operations are extracted from the mentioned work.

1) Horizontal partitioning: *specializes a general entity into multiple different tables depending on their kind. For example, "Documents" are distinguished into "Sales Documents" and "Delivery Documents" with different tables.*

2) Vertical partitioning; *distributes properties of one entity into multiple different tables. For example, the "Changes" to a "Delivery Document" are not stored in the "Delivery Documents" table, but in a separate "Document Changes" table.*

3) Horizontal Anti-Partitioning: *generalizes data from multiple entities into one table. For example, changes of different document types are all stored in the same "Document Changes" table rather than in separate tables.*

4) Vertical Anti-Partitioning *aggregates attributes of multiple entities into the same table. For example, "Sales Documents" aggregates attributes for "Sales Order" and "Return Order" (even though "Reference id" is only required by "Return Order"). The examples also show that one table may be the result of multiple such operations.*

These operations translate well from the previous studied systems (SAP and Oracle) into cloud systems, specially 4) Vertical Anti-Partitioning. Chapter 6 explains the implementation in XTract v3.

The third challenge is related to the level of granularity in the activities names. Granularity of activities can be defined as the amount of detail expressed in the name of an event. For instance, when a change occurs, the name of such activity can be *"Change of X"*, where X is the specific entity that changed. If specific values related to the changes are available, activities could be named *"Change of X to Y"* or *"Change of X from Z to Y"*, where Z and Y are the previous and new values after the change. During this work, we take advantage of the information stored in changes tables to define granularity levels. Section 6.2.1 defines the proposed granularity levels and the implementation in XTract v3.

An open problem in process mining is known as divergence and convergence. In [9] both terms are defined as follows:

Definition 2.5.1 *(Convergence)*

The same activity is executed on multiple process instances at once.

Definition 2.5.2 *(Divergence)*

For one process instance the same activity is performed multiple times

In one of our case studies, the generated event log suffers from convergence. Nevertheless, this is accepted since the activity which occurs in multiple process instances at once provide more information regarding the real process. A possible solution would be to ignore the activity that causes the problem. This is discussed in a more detailed way in Chapter 7.

The last challenge consists of applying process mining techniques (i.e. model discovery, conformance checking, social interactions, etc.) on the generated event log, for which existing process mining tools shall be used.

During this chapter we covered the most relevant concepts in the process mining field. Definitions such as event logs, events, attributes and cases were summarized in Section 2.1. Section 2.2 described the Traditional Approach to produce events logs. The Artifact-Centric Approach was introduced in Section 2.3, while other options to transform the extracted information into event logs were summarized in Section 2.4. Finally, Section 2.5 described the most important aspects when applying process mining to cloud systems, such as the *"Relational Schemas vs High Level Models"* challenge.

Chapter 3

Requirements

The aim of this chapter is to introduce the requirements for event log extraction imposed by process mining as a technology. These elements are related to the extraction of information from cloud systems, the data modelling in such systems and the event logs formatting. In conclusion, this chapter introduces the high-level requirements that will be used to produce a feasible solution. This solution will be explained in chapters 4, 5 and 6. Requirements specifically elicited for the IT company are omitted from this public version of the report.

Data extraction from cloud systems follows a standard procedure as long as these systems provide data access via an API. The concept of API was introduced in Chapter 2. This interface allows for the extraction of information in a structured way. Nevertheless, what kind of information, the amount and the formatting highly depends on the available API. Consequently, all the needed information might not be available and documentation can be sparse. It is also possible that there are different versions of APIs, since companies are constantly updating these interfaces. Until the present date, more than six different versions have been developed and some of these versions are not available anymore[1]. Since different systems handle APIs differently, this work is not able to create a fit for all software tool for cloud systems data extraction. The requirement is to (REQ1)*create an API-specific software tool which obtains all the meaningful information available from the target cloud system.* API-specific software tool will be called API extractor for the rest of the work.

A second requirement comes from the basic nature of process mining technology. In order to apply the multiple process mining techniques developed over time, an event log is needed. Consequently, (REQ2) *a way to transform the extracted data into event logs is required.*

The last requirement from this section consists of the addition of attributes to the activities tables. Event logs can contain more information, for instance, the resource who executed the activity. For this reason, possibility to (REQ3) *add additional attributes to the event log* is needed.

One of the main objectives of the present project is to reduce the amount of time on the transformation step. As it was explained in Chapter 2, the current method to make transformations is by creating and running SQL scripts. The usage of these scripts have several limitations. A person with SQL expertise is needed and they need to create a SQL script for each system. Sometimes, a system holds information for multiple processes, therefore multiple scripts need to be developed. Furthermore, each system requires different amount of time to understand. The goal should be to (REQ5) *reduce the amount of time needed to generate event logs.* Besides reducing the amount of time, another requirement is related to reducing the technical expertise. Consequently, (REQ6) *a solution which allows non-SQL experts to generate event logs* should be provided.

[1] JIRA Software API: https://docs.atlassian.com/jira/

ID	Requirement
REQ1	API extractor to obtain available information from a cloud system.
REQ2	Transform extracted data into activities table.
REQ3	Add additional attributes to the activities table.
REQ4	OMITTED.
REQ5	Reduce the amount of time needed to generate event logs.
REQ6	Solution which allows non-SQL experts to generate event logs.
REQ7	Only one event log can be used as input.
REQ8	OMITTED.
REQ9	OMITTED.
REQ10	Add sorting column to the activities table.

Table 3.1: High-level requirements summary.

Depending on the cloud system, it can hold information for one process or more processes. Even though there are different studies about generation of multiple processes event logs, most of the process mining tools accept one event log as input. Therefore, (REQ7) *Only one event log can be used as input.*

Finally, there is a need to decide the order of activities when they have equal time stamp. This can be solved by adding a (REQ10) *sorting column in the event log.* Figure 3.1 illustrates how sorting looks in an event log.

CASE_ID	ACTIVITY_NAME	EVENT_TIME	Sorting
5002400000e4fKcAAI	Create Case	2016-08-24 14:17:55.000	1
5002400000e4fKcAAI	Change Owner	2016-08-24 14:17:56.000	50
5002400000e4fKcAAI	Send or Receive Email	2016-08-24 14:17:56.000	120
5002400000e4fKcAAI	Change Owner	2016-10-11 09:30:00.000	50
5002400000e4fKcAAI	Create internal Comment	2016-12-20 09:16:12.000	30
5002400000e4fKcAAI	Change Status to Closed	2016-12-20 09:16:15.000	10

Figure 3.1: Sorting Attribute example.

A summary of the requirements presented in this chapter is listed in table 3.1. These general requirements will be used in Chapter 4 to produce a technical design taking into account different databases, technologies and software tools.

Chapter 4

End to End Process Mining for Cloud Systems

The aim of this Chapter is to propose an end-to-end process mining method for analyzing cloud systems, and an architecture to realize a method that satisfies REQ1 - REQ10 from Chapter 3. In particular, for solving the sub-problem of generating an event log from a relational database, we propose to follow the Artifact-Centric Approach. Before naming all the features, it is important to understand various design elements. To start, process mining phases for cloud systems will be presented in Section 4.1. These phases define the procedure at a high level of abstraction. This framework can be used to standardize end to end process mining. Tools can be created to fulfill specialized tasks, generating modular solutions. Secondly, a technical setup is introduced in Section 4.2 that takes into account the objectives of this work, and existing technology described in Chapter 2. Section 4.3 discusses the available stages of data manipulation along the different phases of the method. These stages are called Modification Stages. We evaluate implementation advantages and disadvantages across the Modification Stages along the different tools of the technical setup. Finally, Section 4.4 provides a list of the needed features to produce the expected results at each process mining phase. These features are obtained from the requirements presented in Chapter 3 and take into account the proposed technical setup.

4.1 Process Mining Phases for cloud systems

This section introduces three process mining phases. These phases provide a high-level of abstraction framework that define the end-to-end mechanism. Extraction Phase, Transformation Phase and Analysis Phases can be seen in figure 4.1.

4.1.1 Extraction Phase

The Extraction Phase is the process of obtaining the information from the cloud system. The goal of this step is to store the information in a database that can be accessed without further interaction with the cloud system. It highly depends on the target system, as every vendor is responsible for the creation of their own APIs. User authentication, data selection and data transfer are the key components of this phase.

The existence of APIs to interact with cloud systems opens the door for multiple decisions. These interfaces provide flexibility as data can be manipulated according to specific goals. There

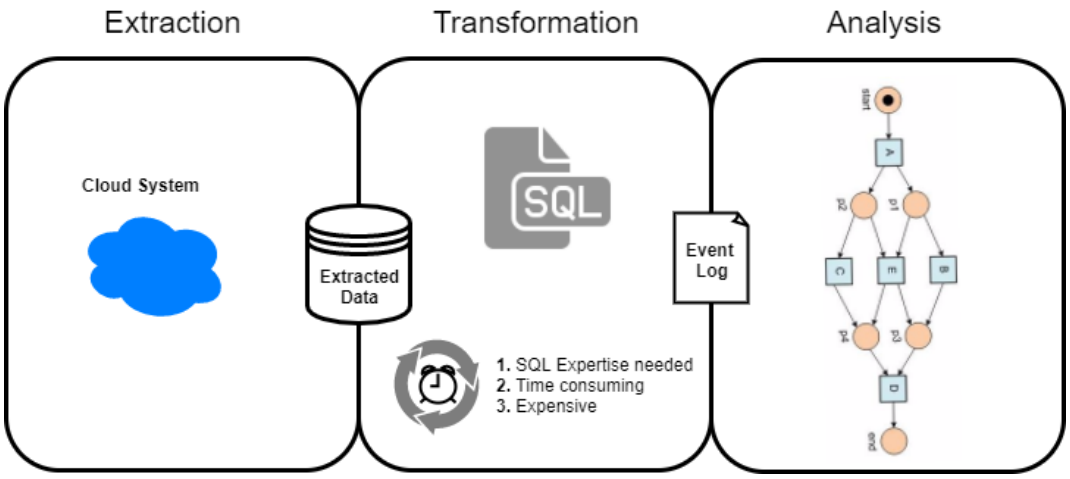

Figure 4.1: Process Mining Phases.

are two approaches when designing API extractors. The first approach refers to extract all the data without taking into account any specific output format. In this case, a NoSQL style would fit well. The second approach consists of adding data processing and manipulation in order to shape the information in a particular data model. The second approach could use a relational database or a graph database as the output format.

In our case, the selected extraction approach uses the second option and a relational database as the result of the Extraction Phase. By extracting the information in a relational database format, less overhead is passed to the Transformation Phase. Furthermore, the Traditional approach introduced in Chapter 2 requires the information to be stored in relational databases. Consequently, we can define the input and output of this phase as follows:

Input: Authentication credentials and API Calls.

Output: Raw data in relational database format.

More information about the designed API extractors will be provided in Chapter 5.

4.1.2 Transformation Phase

The Transformation Phase consists of the set of operations needed in order to construct an event log from the extracted data. The aim of this work is to compare two transformation methodologies. The traditional transformation based on SQL scripts and the automated transformation based on the Artifact-Centric Approach will be compared. Both methods are located in this phase. In Chapter 8 the obtained results are presented. The input and output elements from this phase can be defined as follows:

Input: Obtained relational database information from the Extraction Phase. Independently of which approach is being used, the input is same.

Output: Event Log. This phase ends once an event log is obtained and Process Mining Analysis can begin.

4.1.3 Analysis Phase

In the Analysis Phase different process mining techniques are applied to the obtained event log. The objective of this phase is to obtain insights and meaningful information that was not possible to obtain without process mining. The results generated in this phase demonstrate the real value of process mining. The Extraction Phase and the Transformation Phase can be defined as enablers to make the Analysis Phase possible. The input and output from this phase can be defined as follows:

Input: Event Log.

Output: The output of this phase are the different findings and insights.

4.2 Technical Setup

This section makes the step of creating a technical implementation that permits to fulfill the requirements presented in section 3.1. The proposed technical setup can be observed in figure 4.2. The usage of the Artifact-Centric Approach was a personal choice. The goal of this section is to explain why this setup solves the original requirements while also introducing each component and their interactions.

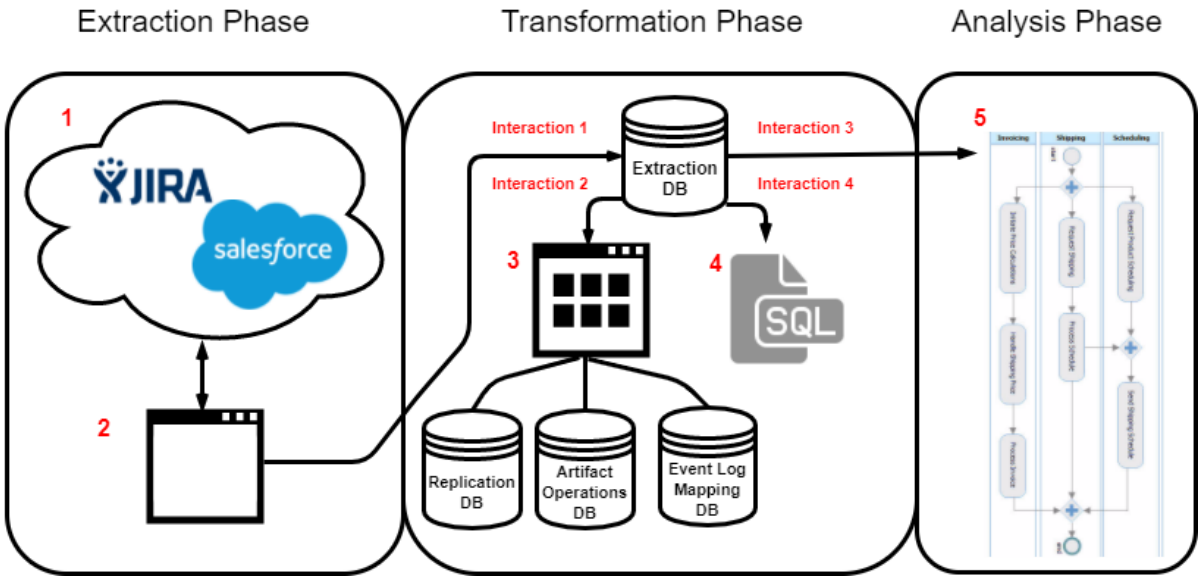

Figure 4.2: Technical Setup.
1: Cloud Systems 2: API Extractor 3: XTract v3
4: SQL Scripts 5: Process Mining Tool

4.2.1 Cloud Systems

In order to validate the obtained results across multiple systems, two cloud systems are used. The first case study is based around Jira. Jira is a cloud system created by Atlassian Corporation. Table 4.1 displays the different Jira applications. Jira Software will be the target system for the first practical demonstration. The second case study is Salesforce. Salesforce stores information for multiple processes. In our case, the selected process is Case Management. Therefore, only the

Name	Description
JIRA Software	JIRA Software is a software development planning tool that provides agile boards, development tool integrations, and reporting for teams using Scrum or Kanban.
JIRA Service Desk	JIRA Service Desk is a collaborative IT service desk with a powerful ticketing system, a self-service knowledge base and real-time reporting.
JIRA Core	JIRA Core is a streamlined issue tracker for business teams. Currently, JIRA Core does not have any application-specific functionality that you need to be aware of. JIRA Core's APIs, plugin modules, and project types are the same as those provided by the JIRA platform. There is no additional documentation for JIRA Core.

Table 4.1: Jira Application's Family

Name	Support Case Management Process
Sales Cloud	Yes
Service Cloud	Yes
Marketing Cloud	No
Community Cloud	No
Wave Analytics	No
App Cloud	No
IoT Cloud	No
Commerce Cloud	No

Table 4.2: Salesforce Application's Family

related objects and information to case management were extracted. Salesforce for Service and Salesforce for Sales support this process. The API Extractor is able to work on both systems as the API calls, objects and design are the same. More specific information about the cloud systems used for the case studies can be found on Chapter 7.

4.2.2 API Extractor

A specific API extractor needs to be developed for each cloud system. The API Extractor used for the first case study was designed and implemented using Java. This component satisfies REQ1. More information about design decisions on the API Extractor can be found on Chapter 5.

4.2.3 SQL Scripts

SQL scripts are the traditional mechanism to generate event logs. Similarly to API Extractor, when using the traditional transformation approach, a specific SQL script needs to be developed for each system. SQL scripts satisfy REQ2, REQ3, REQ10, but fail to satisfy REQ5, REQ6 and REQ7. Consequently, a new component which also satisfies REQ5 and REQ6 should be developed. More information about SQL Scripts can be found on Chapter 2.

4.2.4 XTract

In order to solve the unsatisfied requirements REQ5 and REQ6, XTract is proposed. This software tool is used to validate the Artifact-Centric Approach and is responsible for the generation of the event log in the Transformation Phase. As the existing tool has some limitations, a new version

will be implemented and named XTract v3. This tool satisfies REQ2, REQ3, REQ5, REQ6, REQ7, REQ10. In Section 4.4 a list of required features is presented, while the implementation decisions can be found on Chapter 6.

4.2.5 Process Mining Analysis

Process Mining Analysis will be provided in order to demonstrate to demonstrate the value of process mining and the benefits of generating event logs.

4.2.6 Databases

In figure 4.3 the conceptual design behind the Extraction Database and the Transformation Database can be seen. Although any SQL database can be used, for the case studies MSSQL was selected for the Extraction Database, while HyperSQL was used for the Transformation Database. The Transformation Database is given by the original XTract design and does not suffer any changes. The different databases are introduced because their role is key to understand Modification Stages in Section 4.3.

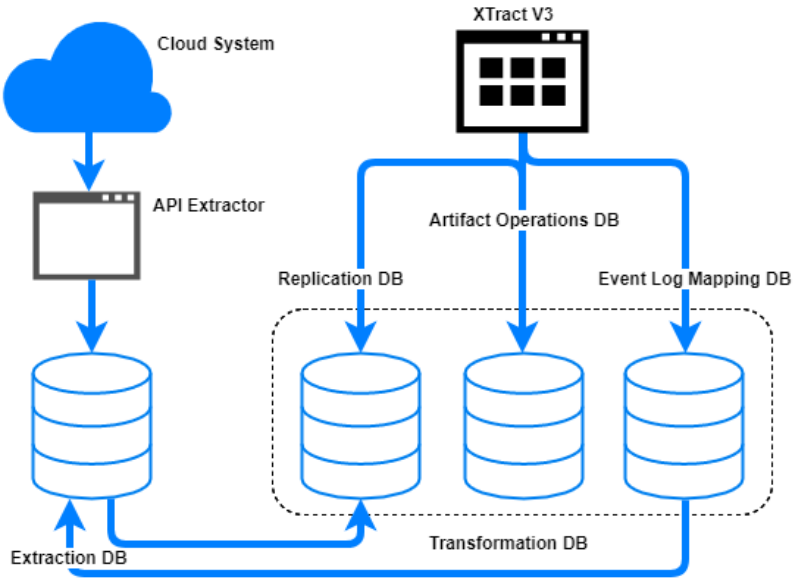

Figure 4.3: Extraction Database and Transformation Database.

Extraction Database

Information regarding the database design is presented Chapter 5. In figure 4.2 there are three components interacting with it. A) API Extractor, B) SQL Scripts (Traditional Approach), C) XTract V3 (Artifact-Centric Approach).

Interaction 1: The Extraction Database stores the data obtained from the API Extractor. The relational database design is created in this component and depends on the cloud system structure.

Interaction 2: For the Artifact-Centric Approach, XTract V3 will connect to this database and replicate the extracted information into the Transformation Database. Once XTract V3 generates

the activities table it will connect and transfer it to the Extraction Database.

Interaction 3: Connection between the transformed data and the process mining tool.

Interaction 4: For the Traditional Approach, the SQL scripts will be deployed in this database. The activities table generated from the script will be stored in this database.

Transformation Database

The technical setup diagram (Figure 4.2)illustrates how XTract V3 is interacting with three different databases. The combination of these databases is known as Transformation Database, as they are only used in the Transformation Phase. They maintain the same functionality and design described in [34]. During the present work, a different naming structure is proposed (Replication Database, Artifact Operations Database and Event Log Mapping Database).

Replication Database (Previously no specific name):

The Replication Database is a duplication generated by XTract V3 from the Extraction Database. The goal of this replication is to avoid data corruption.

Artifact Operations Database (Previously Meta Database):

The Artifact Database is responsible for storing meta data of the Replication Database, metrics for artifact discovery, table references, created artifacts and their structure.

Event Log Mapping Database (Previously Cache Database):

The Event Log Mapping Database is responsible for storing the generated event logs, traces, activities and the separate set of attributes. It contains the same information as the XES file, but in a relational database format.

4.3 Modification Stages

The objective of this section is to present the various stages where new features can be implemented. In Section 4.4 the required new features to meet the presented requirements (Chapter 3) will be determined. In order to comply with all the requirements and being able to transform the data into an event log, twenty-two needed features were found. As it is possible to see in the Technical Setup (4.2), there are many components which provide many implementation possibilities. For this reason, a framework that summarizes the different implementation possibilities needs to be defined. Modification Stages are subcategories of the Process Mining Phases, which define where technical implementations to transform the data along the process mining tool chain should be developed. It is based in the Process Mining Phases, the technical setup, the original XTract design, involved databases and performance considerations. The proposed stages are: A) Extraction Stage, B) Database Stage, C) Artifact Stage, D) Event Log Mapping Stage, E) Event Log Mapping Stage, F) Process Mining Tool Stage. In table 4.3 a summary of the Modification Stages and the Process Mining Phase they are part of is displayed. The Data Source column displays the component that holds the data in the particular Modification Stage.

Extraction Stage

The Extraction Stage modifications are related to any operation applied to the raw data after obtaining it from the cloud system and before inserting it into the Extraction Database. Some examples of these operations are:

1. Ignore non-needed objects or attributes.

Modification Stage	Data Source	Process Mining Phase
Extraction Stage	Extraction Database	Extraction Phase
Replication Database Stage	Replication Database	Transformation Phase
Artifact Stage	Artifact Operations Database	Transformation Phase
Event Log Mapping Stage	Event Log Mapping Database	Transformation Phase
Event Log XES Stage	XES File	Transformation Phase
Process Mining Tool Stage	Event Log and Data Model	Analysis Phase

Table 4.3: Modification Stages, data sources and process mining phases.

2. Add, modify or remove data types from an attribute.
3. Add extra information not included in the cloud system.

Replication Database Stage

The Replication Database Stage modifications are related to any operation applied to the data once it is in the Replication Database. Some examples of these operation are (CRUD operations):

1. Remove a column.
2. Change the name of a table.
3. Create new tables.

Artifact Stage

The Artifact Stage modifications are related to any operation applied to the discovered artifacts structure. Some examples of these operations are:

1. Add a table to an artifact.
2. Remove an activity from an artifact.
3. Modify the main table of an artifact.

Event Log Mapping Stage

The Event Log Mapping Stage modifications are related to any operation applied to the Event Log Mapping Database. Some examples of these operations are:

1. Remove a case.
2. Remove an attribute from a case.
3. Add an activity to a case.

Event Log XES Stage

The Event Log XES Stage modifications are related to any operation applied to XES file. The possible changes are the same as in the previous stage. The difference is that in this case the changes occur in the XES file.

1. Remove a case.

2. Remove an attribute from a case.
3. Add an activity to a case.

Process Mining Tool Stage

The Process Mining Tool Stage modifications are related to any operation applied to data within the Process Mining Software.

1. Change an attribute type.
2. Modify an attribute information.
3. String manipulation of text attributes.

In the following section, the needed features to obtain the target event log with the proposed setup will be introduced. Each feature will be sorted according to the Process Mining Phase where it could be solved. Furthermore, the Modification Stage and the involved databases will be mentioned too.

4.4 Needed Features

In Section 4.1, we have established the Process Mining Phases to provide a high level of abstraction methodology in order to implement end-to-end process mining for cloud systems. Section 4.2 proposed a technical setup consisting of different components such as an API extractor, XTract, SQL scripts and multiple databases. Section 4.3 suggested different areas where the required new features could be developed in the form of Modification Stages. In the current section, a list of specific required features will be presented. These features were gathered by iterative exploration of the data sets, developing intermediate prototypes, conducting several tests on the Traditional Approach and keeping in mind Chapter 3 general requirements.

Tables 4.4, 4.5, 4.6 show all the needed features and in which specific Process Mining Phase they can be solved. The possibility to include a specific feature into a Process Mining Phase depends on several factors. To start, a feature might have no relation to a phase according to their definition. For example, the creation of artifacts should not be handled in the extractor, as this is a step in the Artifact-Centric Approach which occurs in the Transformation Phase. Secondly, it might be possible that the required information to complete the task is not present at the current Modification Stage. For instance, it is not possible to change the name of the activities in the extraction phase because activities have not been discovered at this point of the process. Sections 4.4.1, 4.4.2, 4.4.3 discuss the needed features for the Extraction Phase, Transformation Phase and Analysis Phase respectively. Although advantages and disadvantages of implementing

ID	Feature	Possible Process Mining Phase
A	Information stored in a relational database format	Extraction
B	Ignore non-needed information	Extraction Transformation
C	Changes tables representation	Extraction
D	Domain discovery	Extraction Transformation
E	Database schema identification	Extraction Transformation
F	API Authentication	Extraction
G	Data transferring	Extraction

Table 4.4: Needed features in the Extraction Phase.

the features at different Modification Stages or Process Mining Phases, the selected solutions will be disclosed in Chapter 5 and 6.

4.4.1 Extraction Phase

As it is possible to see in table 4.4, there are some features that should be handled only in the Extraction Phase *(A, C, F, G)*. One of the requirements when using the Artifact-Centric Approach is (A)*information needs to be stored in a relational database format*. The output from the API Extractor needs to be in that format. The next feature is related to changes tables. As it was mentioned in Chapter 2, changes table contain the majority of the activities. Figure 4.4 shows the (C)*changes table representation* options. The first option represents the changes table in one table only. The second option represents the changes table in two tables. The drawback of the first option is that more data will be stored, as some information will be repeated. The drawback of the second option is that an additional relation needs to be created. This means that the approach should be able to detect relevant information that is not directly connected to the main table. The selected solution will have an effect on the Transformation Phase, when artifacts, activities and attributes are discovered. Therefore, the ideal solution should integrate both options.

(F)*Authentication* is a requirement for every cloud system to provide security access. It permits only accredited users to use an API. There are different authentication procedures. Security is not a priority in the current scenario. Consequently, any kind of authentication can be applied. The selection will depend on the available options. (G)*Data transferring* is required in order to transport the information to an instance that can be accessed by the other components. Once again, efficient transportation is not a priority, therefore any transportation mechanism can be used.

There are three features that can be handled at the Extraction Phase or Transformation Phase *(ID: B, D, E.)*. Depending on the cloud system, the API can give access to plenty of information. Parts of this information is never needed for the Transformation Phase and Extraction Phase. Therefore, having the possibility to (B)*ignore non-needed information* at Extraction Stage improves performance and reduces storage size. Non-needed data could also be removed at the Replication Database Stage in the Transformation Phase. The second option is less efficient since non-needed information would be stored in the Extraction Database and in the Replication Database.

There are two possibilities to handle (D)*domain discovery* at the Extraction Stage. Some APIs have the capability to distinguish specific data types for each value. In case this capability is not available, domain discovery has to be implemented. In this scenario, considerable overhead will be added at the Extraction Stage. Domain discovery can also be handled at Replication Database Stage in the Transformation Phase. If this is the case, overhead will be added in the Transformation Phase but not in the Extraction Phase. The last feature is (E) *database schema identification*. The same overhead logic applies, but there is another important factor. Taking into account the work by Xixi Lu [26] and the case studies, it is possible to say that *automatic database schema identification* does not work in all cases. Primary keys and foreign keys automatic detection is an open problem [1].

4.4.2 Transformation Phase

In table 4.5 there are sixteen features, which can be developed at the Transformation Phase. The features that can also be handled in the Extraction Phase were covered in the previous section *(B, D, E)*. The rest of the features will be introduced below *(H, I, J, K, M, N, O, P, Q, S, U)*.

(H)*Internal replication* is needed in order to avoid directly working on the extracted data.

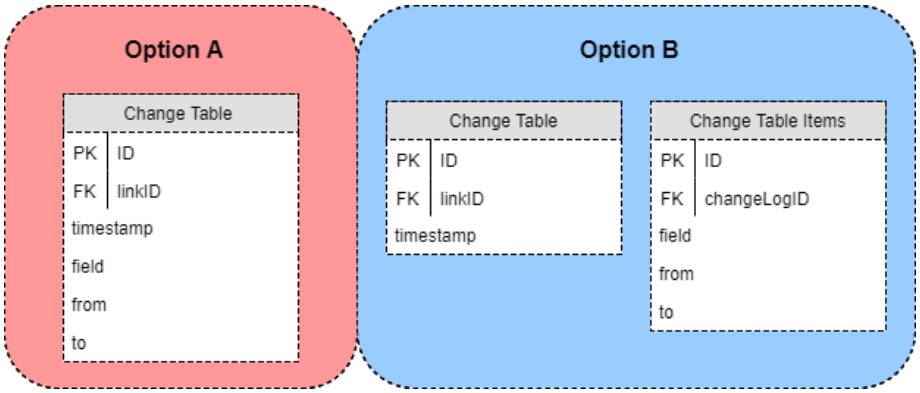

Figure 4.4: Change Table representation options.

ID	Feature Name	Possible Process Mining Phase
H	Internal replication	Transformation
B	Ignore non-needed information	Extraction Transformation
D	Domain discovery	Extraction Transformation
E	Database schema identification	Extraction Transformation
I	Automatic artifact discovery	Transformation
J	Manual artifact discovery	Transformation
K	Activities discovery.	Transformation
M	Attributes discovery.	Transformation
N	Removal of non-needed attributes.	Transformation
O	Removal of non-needed activities.	Transformation
P	Removal of repeated activities.	Transformation
Q	Activity naming	Transformation
R	Addition of attributes	Transformation
S	Activities granularity discovery	Transformation
T	Activities grouping	Analysis Transformation
U	Activities table generation	Transformation

Table 4.5: Needed features in the Transformation Phase.

Replication will be accomplished by duplicating the information from the Extraction Database to the Replication Database. This should be handled at Replication Database Stage. There are two options to discover artifacts. On one hand, (I)*automatic artifact discovery* is the process of discovering one or multiple artifacts without user intervention. On the other hand, (J)*manual artifact discovery* is the possibility to manually select the main table and the rest of the tables that conform an artifact. As it was stated in the requirements (Chapter 3), currently we are looking to generate one event log, therefore the goal is to discover one artifact. Not using the automatic artifact discovery makes the approach semi-automatic, but allows to obtain the target artifact. This is crucial since the end goal is to generate the same event log as with the traditional way. Furthermore, without incorporating manual artifact discovery, the ability to select the right case id will depend on the data model. The ideal solution should be able to offer both possibilities. In previous work, the objective was to discover multiple artifacts [26], [27], but manual artifact creation was also implemented. Both features can only be implemented at the Artifact Stage.

(K,M)*Activities and attributes discovery* are related features. In the case of activities, it is key that all are discovered. Attributes can be added as additional resources on the activities table. Therefore, advanced functionality to manipulate attributes might be needed. Activities and attributes discovery can only be handled at the Artifact Stage.

Editing features are required once an artifact, its activities and its attributes are discovered. (N)*Removal of non-needed attributes* and (O)*non-needed activities* are key in order to produce the exact target event log. They allow to remove information that should not be in the target event log. (P)*Removal of repeated activities* is also required. Figure 4.5 display and example on why there can be repeated activities. For instance, the *CreatedDate* or *ModifiedDate* column values are also stored in the change table. Similarly to removal of non-needed columns and tables, these features also improve performance and reduce storage size. It is possible to develop this functionality at the Artifact Level, Event Log Mapping Level or Event Log XES Level. Ideally, they should be removed at Artifact Level, before the Event Log Mapping is generated. By doing this, the removal will happen before all the traces are discovered, improving performance and reducing storage size.

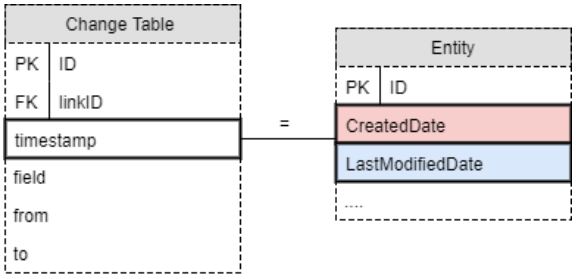

Figure 4.5: Repeated Activities in Cloud Systems.

(Q)*Activity naming* is the possibility to give a specific name to an activity. It adds a layer of customization to the event logs. For instance, event logs can easily be translated to different languages. In previous work, this feature was added by changing the raw data information, which means at Extraction Level or at Replication Database Level [26]. For example, modifying the column names in the Replication Database or adding new columns with activity names. The disadvantage of this method is activity names are introduced before activities are discovered. For this reason, it might be difficult to identify what to change. Another possibility is to change the activity names at Artifact Level, Event Log Mapping Level or Event Log XES Level. The ideal solution consists of changing the activities names at the Artifact Level. The reason is the same as for (N)*Removal of non-needed attributes* or (P)*Removal of repeated activities*.

Static attributes can be defined as information that is linked to an activity and never changes. (R)*Addition of static attributes* are needed to include sorting. Sorting is needed in the cases when different activities happen at the same time (Same time stamp value). Therefore, it is a static attribute because it will always be the same across the different activities. An example was presented in Chapter 3. Static attributes can be included at Database Level, Artifact Level, Event Log Mapping Level or Event Log Level.

(S)*Activities granularity discovery* is the capability to expand or collapse the same activity by adjusting the level of abstraction. Figure 4.6 provides a visual example where the activity *"Change of Status"* is expanded. The same activity can be divided into *"Change Priority to X"* or *"Change Priority from Y to X"*. The needed information is present at the Artifact Level, Event Log Mapping Level and Event Log Level. The disadvantages of implementing the solution at Event Log Mapping Level or Event Log XES Level is that activities are already discovered and all the traces have been generated. Ideally, the activities granularity can be controlled when discovering them at the Artifact Level.

(T)*Activities grouping* is the complete opposite of (S)*activities granularity discovery*. In this case, the objective is to combine two or more activities into one. Once again, this could be developed at Artifact Level, Event Log Mapping Level or Event Log Level. Following the same reasoning provided for previous features, it makes more sense to control the grouping before the event log is generated. Therefore, the best option is to solve this requirement at the Artifact Level. It is also possible to solve the activities grouping at the Process Mining Tool Level. The

CASE_ID	ACTIVITY_NAME	EVENT_TIME
5002400000FpgelAAB	Create Case	2016-02-12 11:36:57.000
5002400000FpgelAAB	Change Owner	2016-02-12 11:36:59.000
5002400000FpgelAAB	Change Status to Working	2016-02-12 13:46:24.000
5002400000FpgelAAB	Change Status to Awaiting Reply	2016-02-12 17:10:23.000
5002400000FpgelAAB	Change Status to On Hold	2016-02-15 15:55:37.000
5002400000FpgelAAB	Change Priority to Low	2016-02-16 11:43:49.000
5002400000FpgelAAB	Change Status to Awaiting Reply	2016-03-01 12:59:48.000
5002400000FpgelAAB	Change Status to On Hold	2016-03-05 10:32:51.000
5002400000FpgelAAB	Change Status to Awaiting Reply	2016-04-25 15:26:25.000
5002400000FpgelAAB	Change Status to Closed	2016-05-05 11:35:26.000

Figure 4.6: Activities granularity example.

ID	Feature Name	Possible Process Mining Phase
S	Activity grouping	Transformation Analysis
X	Model discovery	Analysis
W	Process Mining Analysis	Analysis

Table 4.6: Needed features in the Analysis Phase.

disadvantage of this option is that not all process mining tools are able to group activities, therefore is a less general solution.

The last required feature in the Transformation Phase is the creation of the activities table. (U)*Activities table generation* corresponds to the creation of the mentioned table and exporting options. In addition to case id, activity name and time stamp, other attributes should be added. In this case, the functionality needs to be created after the event log has been generated. Therefore, it could be added at Event Log XES Level or Event Log Mapping Level. The drawbacks of implementing it at Event Log XES Level is that a XES file parser needs to be implemented. Querying the Event Log Mapping Database is more efficient than interacting with the XES file.

4.4.3 Analysis Phase

Table 4.6 contains the three features that are needed in the Analysis Phase *(ID: S, X, W)*. As (S)*activities grouping* was introduced previously, this section focuses on the remaining two features.

The last two features can be handled by process mining tools as they can handle (X)*model discovery* and are specialized in (W)*process mining analysis*.

During this chapter, we defined the process mining phases and designed a technical setup considering these phases and the project objectives. Section 4.3 introduced the concept of Modification Stages, which are the different areas along the procedure where needed features could be implemented. Finally, in Section 4.4 a list of the needed features was displayed. An explanation on why these features are needed was shown. The selected design solutions for the Extraction Phase will be discussed in Chapter 5 and Chapter 6 covers the Transformation Phase.

Chapter 5

Extractor Design and Implementation

The goal of this chapter is to illustrate the API Extractor design decisions and implementation. Therefore, this chapter focuses on Extraction Phase of the procedure outlined in Section 4.1. It takes into account the requirements from Chapter 3 and the needed features from Chapter 4. Section 5.1 makes a detailed description about the API extractor operation. For each needed feature described in Chapter 4 (figure 4.4) the selected solution will be presented in Section 5.2. The decisions are based on the Jira Software [1] and Salesforce API [2]. Jira Software API extractor was created from scratch, while a Salesforce API extractor was re used. Both API extractor maintain a very similar structure.

5.1 API Extractor Operation

In figure 5.1 the different components of the API extractor and the external entities can be seen. The API extractor components are represented in four blocks, while the external entities are the cloud system and the Replication Database. The role of each block will be discussed in this section and linked to the specific feature in Section 5.2.

External Components

The cloud system and the Replication Database need to be taken into account when designing the API Extractor. On one hand, the cloud system is fundamental as its implementation affects the general design. Elements such as the authentication mechanism, availability of information, relational database design and the underlying data inter exchange technology depend strictly on the cloud system interface. On the other hand, the desired Replication Database technology needs to be implemented in the API Extraction by selecting the correct database driver. MSSQL will be used as the Replication Database.

The different parts of the API extractor are divided into blocks with specific tasks. The following explanation is based on the Jira Software API Extractor as it was created from scratch. Nevertheless, differences between this extractor and the Salesforce extractor will be mentioned.

1. Cloud System Credentials

This block is responsible for the user authentication. As it was mentioned before, the authen-

[1] JIRA Software API: https://docs.atlassian.com/jira-software/REST/server/
[2] Salesforce API: https://developer.salesforce.com/docs/atlas.en-us.api.meta/api/sforce_api_objects_list.htm

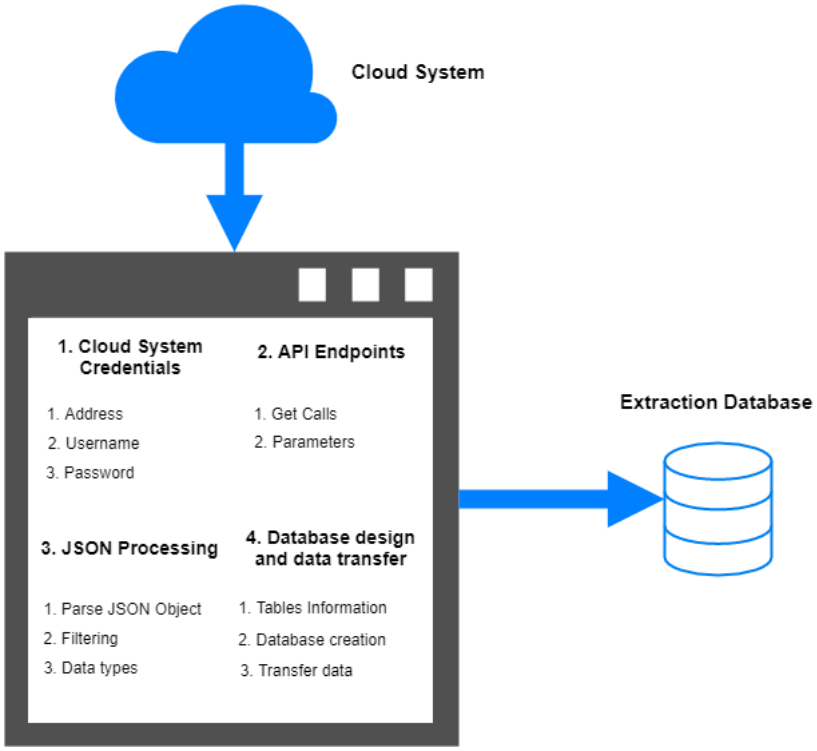

Figure 5.1: API extractor overview.

tication mechanism depends on the options provided by the cloud system. Basic Authentication was used for both extractors. Consequently, only user name, password and address is needed. This authentication mechanism reduces overhead compared to other available options.

2. API Endpoints

An API Endpoint can be defined as the interface that enables the transmission of information between the API Extractor and the cloud system. In Rest APIs, this channel has the form of an URL where an object data is available. Furthermore, different parameters might be available to shape the desired response in order to reduce or increase the desired information. In this block the desired API calls are programmed and made available. For our case studies, only the desired endpoints were created, while endpoints which returned non-needed information were excluded. For the first case study, a library [3] was used to reduce development time. This library handled the available endpoints, parameters and responses. In the case of Jira Software API, the endpoints produce a response in JSON format.

Call Request: https://jira/rest/api/2/project


```
[{"expand":"description,lead,url,projectKeys",
"self":"https://jira/rest/api/2/project/10502",
"id":"10502",
"key":"XXX1",
"name":"Name1",
"avatarUrls":{"48x48":"https://jira/secure/projectavatar?avatarId=10903",
"24x24":"https://jira/secure/projectavatar?size=small&avatarId=10903",
"16x16":"https://jira/secure/projectavatar?size=xsmall&avatarId=10903",
"32x32":"https://jira/secure/projectavatar?size=medium&avatarId=10903"},
"projectTypeKey":"typeKey1"},
{"expand":"description,lead,url,projectKeys",
```


[3] rcarz / jira-client: https://travis-ci.org/rcarz/jira-client

```
"self":"https://jira/rest/api/2/project/10301",
"id":"10301",
"key":"XXXXX",
"name":"Project Name 1",
"avatarUrls":{"48x48":"https://jira/secure/projectavatar?pid=10301&avatarId=10011",
"24x24":"https://jira/secure/projectavatar?size=small&pid=10301&avatarId=10011",
"16x16":"https://jira/secure/projectavatar?size=xsmall&pid=10301&avatarId=10011",
"32x32":"https://jira/secure/projectavatar?size=medium&pid=10301&avatarId=10011"},
"projectTypeKey":"software"},
{"expand":"description,lead,url,projectKeys",
"self":"https://jira/rest/api/2/project/10200","id":"10200",
"key":"XXX2",
"name":"Project Name 2",
"avatarUrls":{"48x48":"https://jira/secure/projectavatar?pid=10200&avatarId=10011",
"24x24":"https://jira/secure/projectavatar?size=small&pid=10200&avatarId=10011",
"16x16":"https://jira/secure/projectavatar?size=xsmall&pid=10200&avatarId=10011",
"32x32":"https://jira/secure/projectavatar?size=medium&pid=10200&avatarId=10011"},
"projectTypeKey":"typeKey2"}]
```


Listing 5.1: API response.

In listing 5.1 an anonymized example of the JSON response is shown after executing the API endpoint call. Since no parameters were included in the API call, the reply will contain all the projects available in the system.

3. JSON Processing

The mentioned library contains all the available Jira Software objects in Java classes, which provides an efficient mechanism to transform the JSON payload into Java objects. Once the information is stored in Java objects, it is possible to map it to the desired relational database design. The actual mapping to the database occurs in the last block. The JSON processing block is also responsible for filtering non-needed attributes that were not filtered with the endpoint calls. Finally, data types detection can be handle in this step. For our case, the library contained the required data types already defined in the Java object attributes. In the case of Salesforce, the API shares the specific objects data types.

```
public class Project extends Resource {

    private Map<String, String> avatarUrls = null;
    private String key = null;
    private String name = null;
    private String description = null;
    private User lead = null;
    private String assigneeType = null;
    private List<Component> components = null;
    private List<IssueType> issueTypes = null;
    private List<Version> versions = null;
    private Map<String, String> roles = null;
    private ProjectCategory category = null;
    private String email = null;

    /**
     * Creates a project from a JSON payload.
     *
     * @param restclient REST client instance
     * @param json JSON payload
     */
    protected Project(RestClient restclient, JSONObject json) {
        super(restclient);

        if (json != null)
            deserialise(json);
    }
```

Listing 5.2: Java Object example.

In listing 5.2 an example object can be seen. In this case, the object represents a Project. The requirement is that the attributes follow the same format as in the JSON payload. This format is related to the type of structure that holds the information. For instance, the usage of Java Array, Java Map or another structure needs to match the JSON payload structure. Furthermore, it is possible to define the data types of each element, which can be used when creating the relational database.

```
public List<Object[]> processProjects(List<Project> projects) {

  List<Object[]> outputData = new ArrayList<>();

    for(int j = 0; j<projects.size(); j++){
        Project project = projects.get(j);
        String[] columns = tableMetaData.getColumnsArray();
        Object [] currentValuesSelected = new Object [tableMetaData.
            getColumnMetaData().size()];
        for(int i = 0; i < columns.length; i++){
            for(int x = 0; x<currentValuesSelected.length; x++){
                currentValuesSelected[i]=null;
             }
             if(columns[i].equalsIgnoreCase("id")){
                if(project.getId()!=null){
                    currentValuesSelected[i]=Integer.parseInt(project.getId());
                }
            }else if(columns[i].equalsIgnoreCase("description")){
                if(project.getDescription()!=null){
                    currentValuesSelected[i]=project.getDescription();
                }
            }else if((columns[i].equalsIgnoreCase("name"))){
                if(project.getName()!=null){
                    currentValuesSelected[i]=project.getName();
                }
            }else if((columns[i].equalsIgnoreCase("key"))){
                if(project.getKey()!=null){
                    currentValuesSelected[i]=project.getKey();
                }
            }else if((columns[i].equalsIgnoreCase("assigneeType"))){
                if(project.getAssigneeType()!=null){
                     currentValuesSelected[i]=project.getAssigneeType();
                }
            }else{
                //Something wrong in the processConfiguration about table fields
                currentValuesSelected[i]="ERROR";
            }
        }
        outputData.add(currentValuesSelected);
    }

    return outputData;

}
```

Listing 5.3: Processing objects.

In listing 5.3 an example of the method which parses the Project object is shown. Only the relevant attributes are selected.

4. Database design and data transfer

In the last block, the database design and transfer of information is handled. In order to create the database, the table names, the column names, and the data types should be defined. Listing 5.4 shows the function were the selected tables are defined.

```
public ProcessConfiguration(ProcessType processType) {
    this.processType = processType;
```

```
    if(processType.equals(ProcessType.ISSUES)){
        this.processName = "Issues";
        this.tables = new String[]{
                "Projects",
                "Issues" ,
                "Resolutions",
                "IssueTypes",
                "Priorities",
                "FixVersions",
                "Status",
                "Comments",
                "AffectsVersions",
                "Users",
                "ChangeLogs",
                "ChangeLogItems",
                "Attachments",
                "Boards",
                "Sprints",
                "Sprints_Issues"
        };
    }
}
```

Listing 5.4: Tables selection.

The final task that needs to be full-filled in this block is the transmission of data into the Replication Database. A specific database connector was used for this purpose. With this driver, different operations can be done, such as definition and creation of tables, columns and data types. Furthermore, data is transferred to the Replication Database in this block.

5.2 Refinement of needed features in the Extraction Phase

In this section, the presented solution for the Extraction Phase features will be discussed and linked to the specific block that solves the feature from figure 5.1.

A) Information stored in a relational database format

For both of the case studies, each system presented the information organized into individual objects. Each object contains its own identifier and it is clearly linked to other objects. Therefore, the extracted data can be converted into a relational database. Nevertheless, the relational database design and elements mapping needs to be handle correctly. This feature is handled between block 3 and 4. In the JSON processing block, the needed values are stored into Java objects and mapped to the relational database tables.

B) Ignore non-needed information

In order to ignore non-needed information only the needed API Calls should be used. Depending on the level of sophistication of the API, multiple options to filter the endpoints responses could be available. For our first case study, the API has advanced search capabilities. With this option, it is possible to create very specific API Calls. This solution is included in block 2.

If advanced search capabilities are not present, another approach is to ignore non-needed information when mapping the attributes into the relational database. Only the relevant information should be taken into account when designing the relational database. This solution is included in block 4.

C) Changes table representation

As it was mentioned in Chapter 2, changes tables store all the changes that occur in the system.

For that reason, these tables constitute the main source for obtaining activities. Consequently, support for both options was added to the API extractor. The changes tables design and mapping is handled between blocks 3 and 4.

D) Domain discovery

Domain discovery can be handled successfully while defining each individual Java object as it was mentioned in Section 5.1.

E) Database schema identification

Despite the database schema could have been added at the Extraction Level, the decision was to handle this feature in XTract v3. The main reason is to avoid overhead in the API Extractor. Furthermore, having the possibility to identify database schema at the Transformation Phase is a more general solution. Otherwise, every API Extractor should incorporate this feature.

F) API Authentication

The selected method for authentication is HTTP Basic Authentication over cookie based and OAuth. The disadvantage of this authentication option is that is less secure. In any case, security is not a main concern for the current project.

G) Data transferring

Possibility to transfer data into specific database by providing the correct credentials was created. For the first case study the data is transferred to the Extraction Database using JDBC Drivers. MSSQL Database was selected as it is supported by XTract.

This chapter explained the design and operation of the API Extractor. Jira Software API extractor was created, while Salesforce API was re used. Both API extractors are very similar regarding their operation. The main difference between both systems is Salesforce interface shares the data type of the different attributes, while in Jira Software this information is not explicitly shared. Consequently, data types for the attributes can be obtained with an API request in Salesforce, while in Jira they need to be discovered.

Chapter 6

Event Log Extraction Using the Artifact-Centric Approach

This chapter covers the Transformation Phase described in 4.1. Chapter 5 discussed the first challenge of extracting data from a cloud system. An API extractor to transfer the data into a local relational database was created. The next challenge consists of transforming the extracted data into an event log. For this purpose, the Artifact-Centric Approach was selected. Prior work and tools using this approach were tested and adapted. Section 6.1 evaluates existing implementations of the Artifact-Centric Approach to log extraction. This evaluation will help to understand the required changes and extensions. Section 6.2 describes the conceptual and technical realizations of the extensions. All the extensions and changes described in this Section have been implemented and contributed to the open-source project XTract in its version XTract V3.

6.1 XTract V1 Evaluation

To understand sections 6.1 and 6.2, the difference between the Artifact-Centric Approach and XTract should be clear. The former is the conceptual technique, while the latter is the tool used as a proof of concept. Xtract V1 refers to the program created by Erik H. J. Nooijen[34], XTract V2 refers to the program created by Xixi Lu [26] and XTract V3 corresponds to the program created for this project.

In order to evaluate XTract V1, multiple tests were conducted. Different demo and real data sets were available for this purpose. Also, scenarios presented in previous work were recreated when possible. Observations about XTract V1 presented in [26] and [27] were analyzed in detail. From these tests, it was possible to determine which needed features were already implemented and working correctly. Tables 6.1 and 6.2 display the features from table 4.5 XTract V1 is able and not able to accomplish.

6.1.1 Features XTract V1 is able to accomplish

(H) *Internal replication*, (D) *domain discovery* and (M) *attributes discovery* are completed successfully with XTract V1. The same is partially true for (I) *automatic artifact discovery*, as in the first case study the target main table was discovered, but not in the second case. Nevertheless, automatic artifact discovery will always select the most important table as the main table of the artifact. This might not always be ideal, depending on the target event log. For instance,

ID	Feature
H	Internal replication
B	Domain discovery
I	Automatic artifact discovery
M	Attributes discovery

Table 6.1: Features XTract V1 is able to accomplish

ID	Feature
B	Ignore non-needed information
E	Database schema identification
J	Manual artifact discovery
K	Activities discovery
N	Removal of non-needed attributes
O	Removal of non-needed activities
P	Removal of repeated activities
Q	Activity naming
R	Addition of static attributes
S	Activities granularity discovery
T	Activities grouping
U	Activities table generation

Table 6.2: Features XTract V1 is *not* able to accomplish

another table should be selected as the main table in order to use a different case id. For this reason, manual artifact discovery should be developed. As it was mentioned in Chapter 5, domain discovery was also handle at the Extraction Phase. Therefore, the API Extractor and XTract V3 accurately handle this feature.

6.1.2 Features XTract V1 is not able to accomplish

In table 6.2 there are twelve features that are either not implemented, not producing the expected results or not considered in XTract V1.

B) Ignore non-needed information

In order to ignore data, XTract V3 should be able to interact with the Replication Database to remove specific columns or tables. This feature was not implemented previously.

E) Database schema identification

Database schema identification as needed for the Extraction Phase can be divided in two sub-features. The first one is primary key identification. In this case, the obtained results with XTract v1 are not always accurate. This is very notorious in the cases when trying to detect primary keys made of more than one column. The second sub-feature is foreign keys identification. Similarly to the primary keys situation, foreign keys identification fails in many cases. Even thought many foreign keys relations are identified, some of them are incomplete and some are not detected [26].

J) Manual artifact discovery

Manual artifact discovery is implemented in XTract V1. Nevertheless, while working in the case studies, it was found that the manual identification has a flaw. Only tables directly connected with a foreign key to the selected main table will be added to the artifact. It is important that tables not directly connected to the main table can also be included in an artifact.

K) Activities discovery

This feature can be considered as the most important from the list. The challenge when discovering activities is related to how information is stored in relational databases. This challenge was presented in Section 2.5 when *"Relational Schemas vs High-Level Models"* idea was introduced. Furthermore, changes tables play a key role in the discovery of activities. These tables were introduced in Chapter 2. In Chapter 5, it was mentioned that both forms of changes tables will be supported. In figure 6.1 a graphical representation of the challenge is shown. *"Vertical Anti-Partitioning"* causes attributes of multiple entities into the same table. With XTract V1 all the activities included in these tables will be discovered as one event type. In the original XTract design this challenge was not addressed, therefore was not considered.

Id	IsDeleted	CaseId	CreatedById	CreatedDate	Field	OldValue	NewValue
01724...	0	500240000...	0052400000...	2016-02-15 10:...	Contact		
01724...	0	500240000...	0052400000...	2016-09-26 16:...	Contact		
01724...	0	500240000...	0052400000...	2016-06-06 14:...	Contact		
01724...	1	500240000...	0052400000...	2017-04-06 11:...	Contact		
01724...	1	500240000...	0052400000...	2017-04-06 11:...	Contact		
01724...	0	500240000...	0052400000...	2016-02-02 17:...	CurrencyIsoCode		
01724...	0	500240000...	0052400000...	2016-06-17 18:...	Description		
01724...	0	500240000...	0052400000...	2016-06-17 18:...	Description		
01724...	0	500240000...	0052400000...	2016-06-01 18:...	Description		
01724...	0	500240000...	0052400000...	2016-06-01 18:...	Description		

Figure 6.1: Multiple objects represented in the same table example.

N) Removal of non-needed attributes and O) Removal of non-needed activities

Once an artifact, its activities and attributes are discovered, it is not possible in XTract V1 to remove the elements that are not relevant. For that reason, the original version will produce an event log with all the activities and attributes. The target event log should not contain all the activities, since many of them are not relevant. This feature was not implemented previously.

P) Removal of repeated activities

Repeated activities in cloud systems were introduced in Chapter 4 (figure 4.5). The same logic as for non-needed activities applies to repeated activities. This feature was not considered in XTract V1.

Q) Activity naming

It is not possible to customize activities names in XTract V1. The names used in the event logs come from the database tables and columns that hold the time stamp. This nomenclature can be hard to understand. This feature was not implemented previously.

R) Addition of static attributes

This feature was not considered in XTract V1.

S) Activities granularity discovery

This feature was not considered in XTract V1.

T) Activities grouping

This feature was not considered in XTract V1.

U) Activities table generation

In XTract V1, the output was the event logs in the form of a XES file. There was no need to generate an activity table, therefore this feature was not implemented.

6.2 Artifact-Centric Approach Additions

In the following section, we discuss how we extended the Artifact-Centric Approach and XTract V1 to realize the missing features that were discussed in 6.1.2. The specific Artifact-Centric Approach original steps where a change was made will be mentioned. The Modification Stage where the addition is developed as described in Section 4.3 will also be mentioned. Figure 6.2 displays the original Artifact-Centric Approach.

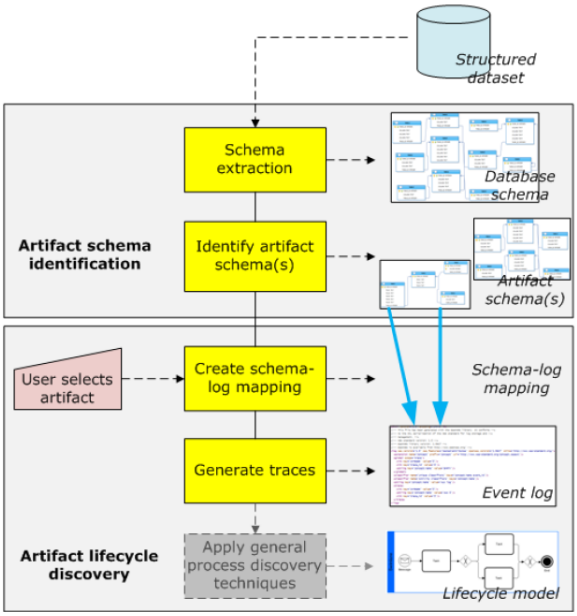

Figure 6.2: Artifact-Centric Approach original method.

6.2.1 Data Preparation

The objective of this step is to add editing capabilities on the extracted information. With the right set of operations, the information can be transformed in a way that the Artifact-Centric Approach can create the target event log. The Data Manipulation step solves the following needed features: (B) *ignore non-needed information* , (K,S)*activities discovery and activities granularity discovery.* All these features were implemented at the Database Replication Stage (Section 4.2.6).

Database Editing Functions

XTract V3 is able to remove non-needed tables and columns, therefore implementing (B) *ignore non-needed information* feature. This feature improves performance and reduce storage size. XTract V3 provides a graphical interface where users can select the tables and columns that should be dropped. Once a table or column is selected, the query generation will be handled automatically. Figure 6.3 shows the user interfaces.

Handling Vertical Anti-Partitioning

As it was mentioned previously, the main issue when discovering activities in XTract V1 is the effect of vertical anti-partitioning (Introduced in Chapter 2). Figure 6.1 displays an example of

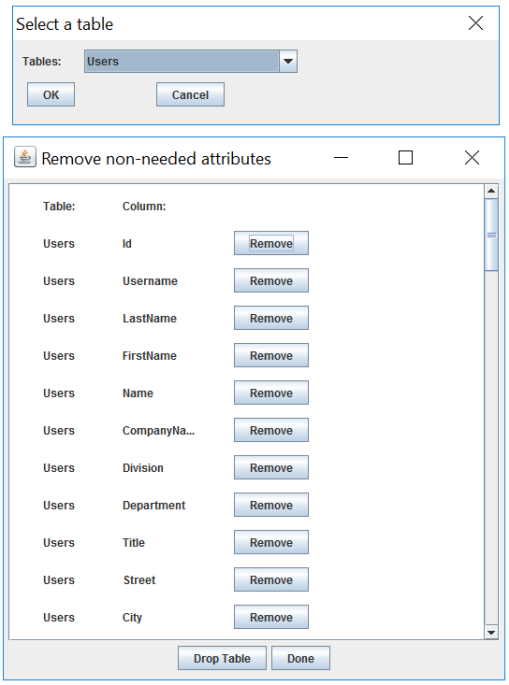

Figure 6.3: Interfaces to remove tables or columns.

a table containing multiple entities. *"Splitting"* tables into individual tables per each entity will solve the *activities discovery* problem. In Chapter 4 (figure 4.4), the representation options of changes tables can be found. For both cases, the field column can be used to split the table into multiple tables.

Definition 6.2.1 *(Field Column) Field column can be defined as the column that holds the object or entity that changed. The column name may be different from one system to another.*

During the case studies, it was found that it is possible to have systems with multiple changes tables (i.e. four changes tables represented with Option 1, or two changes tables represented with Option 2). Therefore, XTract V3 can split multiple changes table represented with either Option 1 or Option 2. In the case of Representation Option 2, a join operation needs to be executed before. Listing 6.1, displays the algorithm to join these tables. The input needs to be provided by the user. The algorithm to solve vertical anti-partitioning is presented in listing 6.2. This algorithm also contains the definition of granularity levels discussed in the next section.

```
1  def joinChangesTables():
2
3    var changesTable1    #First Table
4    var changesTable2    #Second Table containing "From" and "To" Columns
5    var key1             #Reference key from changeTable1 connecting both tables
6    var key2             #Reference key from changeTable2 connecting both tables
7
8    var mergedTable      #Result of joining both tables
9
10   mergedTable = JOIN(changeTable1 AND ChangeTable2 ON Key1=Key2)
11
12   return mergedTable
```

Listing 6.1: Joining changes tables pseudo code (Representation Option 2).

```
def splitTable():

  var tableToSplit           #Table with Vertical Anti-Partitioning
  var fieldColumn            #Field column
  var fromColumn             #Column holding previous value
  var toColumn               #Column holding new value

  var highGranularityFields  #Store user selections
  var lowGranularityFields   #Store user selections
  var noGranularityFields    #Store user selections

  var newTablesList          #New table results

  for value in fieldColumn:
   if(value is selected by user as High Granularity)
     highGranularityFields.add(value)

   if(value is selected by user as Low Granularity)
     lowGranularityFields.add(value)

   if(value is selected by user as No Granularity)
     noGranularityFields.add(value)

  for field in highGranularityFields:
     newTable.setName(S_field_FROM_fromColumn_TO_toColumn)
     newTable =(INSERT ALL WHERE tableToSplit.field=field FROM tableToSplit)
     newTablesList.add(newTable)

  for field in lowGranularityFields:
     newTable.setName(S_field_TO_toColumn)
     newTable =(INSERT ALL WHERE tableToSplit.field=field FROM tableToSplit)
     newTablesList.add(newTable)

  for field in noGranularityFields:
     newTable.setName(S_field)
     newTable =(INSERT ALL WHERE tableToSplit.field=field FROM tableToSplit)
     newTablesList.add(newTable)

  return newTablesList
```

Listing 6.2: Split tables and granularity levels definition pseudo code.

Defining Granularity Levels for Activities

An advantage of changes tables is that they store very detailed information about what changed. The *from* and *to* columns can be used to detect the previous value and the new value. This information is needed to establish the granularity level.

Definition 6.2.2 *(From column) From column can be defined as the column that holds the value before the change occurred.*

Definition 6.2.3 *(To column) To column can be defined as the column that holds the value after the change occurred.*

Granularity Level Name	Columns Used	Example
No Granularity	None	Change Status
Low Granularity	To	Change Status to Done
High Granularity	From	Change Status from Open to Done

Table 6.3: Proposed Granularity Levels.

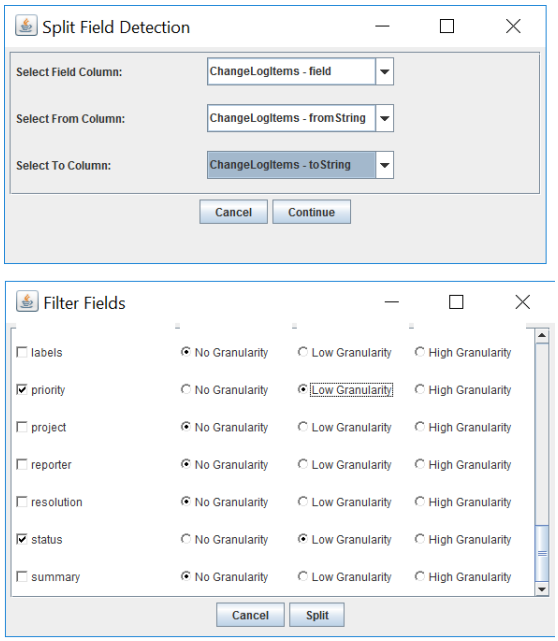

Figure 6.4: Granularity selection.

Using these columns, it is possible to create three granularity levels for naming activities. The proposed levels can be seen in table 6.3. Combining the present idea with the split by field functionality it is possible to create (S) *activities granularity discovery* feature.

To define the granularity levels for activities, XTract V3 provides a graphical interface where the *from*, *to* and *field* columns need to be defined. With this information, XTract V3 will produce all the fields that were changed. The user is then able to choose the fields he is interested in and the granularity level for each field. Depending on this selection, tables for all the possible options will be created.

Figure 6.4 reproduces an example of splitting tables with granularity level. The names of these tables and columns can be different between systems. In the given example, the changes table is split into multiple table containing all the information from priorities and statuses only. Since

New table name
S_status_TO_Todo
S_status_TO_Open
S_status_TO_Released
S_priority_TO_High
S_priority_TO_Critical
S_priority_TO_Low

Table 6.4: Example of naming convention after a split.

granularity was defined as "Low" for both fields, the generated tables will be named according to their values. In table 6.4, a possible output of the split is shown. XTract V3 includes a "S" in front of the tables that were created from a split operation. After this split operation, the activities from change tables are stored in a suitable way to be discovered in later steps.

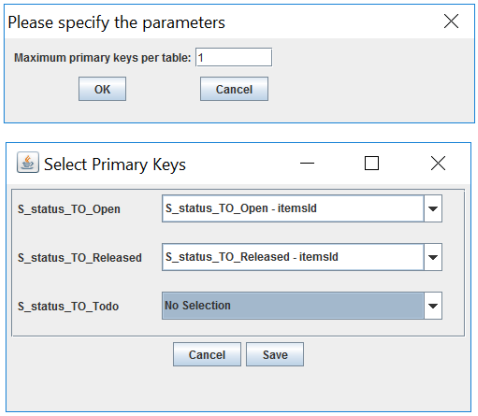

Figure 6.5: Manual primary keys selection.

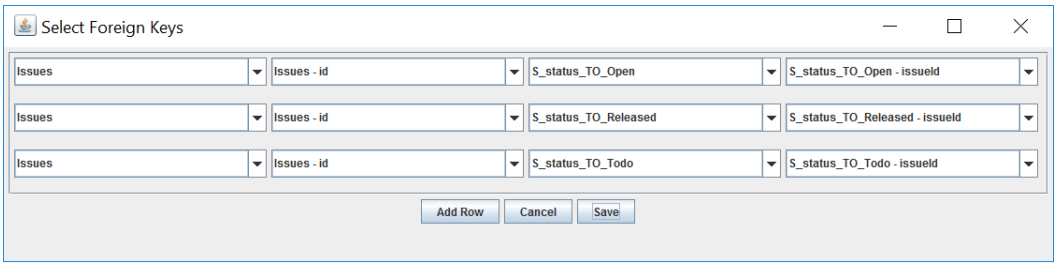

Figure 6.6: Manual foreign keys selection.

6.2.2 Manual Database Schema Identification

Manual database schema identification is required as the automatic *database schema identification* does not produce the expected results. This was noticed in [26], but also validated while working in the case studies. Consequently, manual identification of primary keys and foreign keys were implemented in XTract V3. In figures 6.5, 6.6 manual primary keys and foreign keys identification are shown respectively. Both mechanisms were implemented at the Replication Database Stage.

It is possible to use automatic discovery and manual discovery of primary and foreign keys together. For instance, using the automatic discovery methods implemented in XTract V1 and complete the missing keys manually. Both mechanism store the primary and foreign keys in the Artifacts Operations Database introduced in Section 4.2.6.

6.2.3 Artifact Manipulation

The aim of this step is to provide editing capabilities on the discovered artifacts, its activities and its attributes. It should be included as part of the Artifacts Schema Identification step in the Artifact-Centric Approach. The proposed steps solve the following needed features: (J)*Manual artifact discovery*, (N)*removal of non-needed attributes*, (O)*removal of non-needed activ-*

ities, (P)*removal of repeated activities*, (Q)*activity naming*, (R)*addition of static attributes* and (T)*activities grouping*. All these features were implemented at the Artifacts Stage.

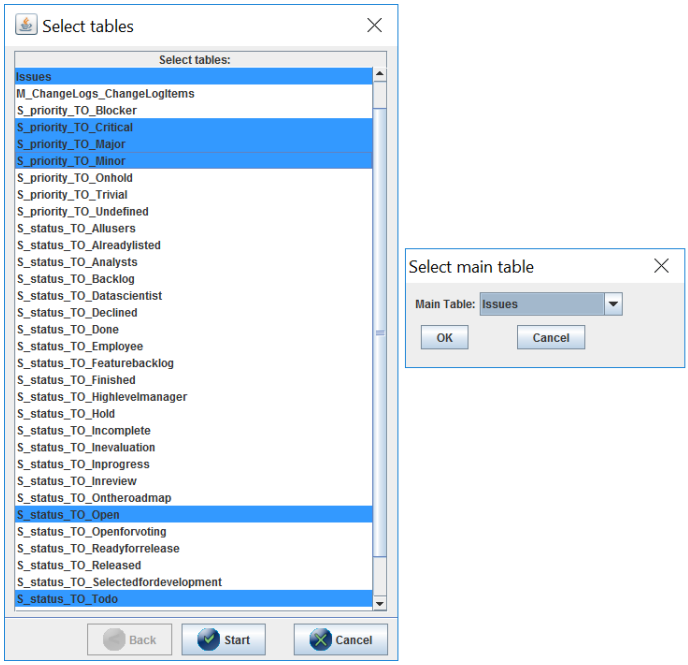

Figure 6.7: Main table and secondary tables selection.

Artifact Creation

As it was mentioned when identifying the features XTract V1 was not able to accomplish, *manual artifact discovery* was not producing the right results. Tables not directly connected to the main table were not added to the artifact. XTract V3 has the possibility to manually create an artifact including tables that are not directly connected to the main table. The user can pick the main table and all the tables associated with the artifact. Figure 6.7 displays the tables selection.

This method will "force" the selected tables to be part of the artifact, without reviewing that there is an actual connection between them. Consequently, the user needs to be sure there is a connection, otherwise unexpected problems can happen in later stages. With the new manual artifact creation, there are three ways to discover artifacts. Automatic artifact discovery and manual artifact discovery from XTract V1 are included in XTract V3.

Besides manual creation of an artifact, changing the main table of an already created artifact was implemented. This function provides additional benefits as different case ids can be picked. More information about additional benefits of changing case ids will be discussed in Chapter 9.

Artifact Editing Functions

Several functions were implemented to modify an artifact once its activities and attributes are discovered. To start, (N, O)*Removal of activities and attributes* is possible in XTract V3. Since any activity can be removed this capability solves the removal of (P)*repeated activities* too.

The new version provides a graphical interface, which summarizes all the activities found in an artifact. These activities are obtained from the Artifact Operations Database (Introduced in

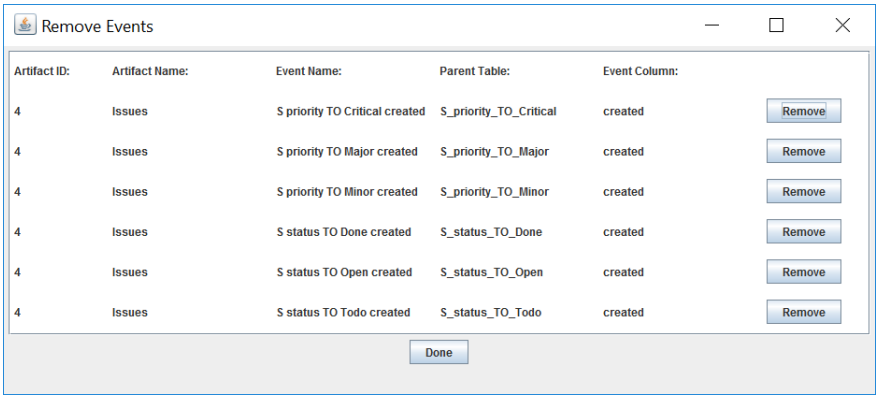

Figure 6.8: Remove activities interface.

Section 4.2.6). The user can select an activity and XTract V3 will handle the queries to remove it from the artifact. By doing this, those activities will not be taken into account in the event log mapping step, excluding them from the traces. The same procedure applies for attributes. In figure 6.8, an example of the interface is presented.

(Q)*Activity names customization* is included by providing an interface where the users can change the names of specific activities. Since there is no limitation on activities names, it is possible to name multiple activities equally resulting in (T)*activities grouping*. Similarly to removal of activities, XTract V3 interacts with the Artifacts Operations Database to update the activity name field. The SQL query is automatically created and executed. The same procedure applies for re-naming attributes.

(R)*Addition of static attributes* is also included by providing the possibility to add attributes of any data type (i.e. integer, string, time stamp) into the activities. Whenever this feature is used, the same attribute (and its value) will be added to selected activities. Listing 6.3 displays the pseudo code of the algorithm. It interacts with the Event Log Mapping Database introduced in Chapter 4.

```
1  def AddStaticAttributes():
2
3    var attributeName      #New attribute name
4    var attributeType      #New attribute data type
5    var newValuesList      #New values for each event type
6    var eventTypeList      #All the event types of the current artifact
7    var dataBaseInstance   #Event log mapping database instance
8
9    eventTypeList = dataBaseInstance.getEventTypes()
10
11   for eventType in eventTypeList:
12
13     eventType.addNewAttribute(attributeName, attributeType, newValueList.get(eventType))
14     dataBaseInstance.update(eventType)
```

Listing 6.3: Add static attributes pseudo code.

6.2.4 Export Activities Table

This step is full filling two goals. The first goal is to generate (U)*activities table*. Activities table consists of a case id, activity name and time stamp, but also includes additional attributes.

XTract V3 will generate the basic activity table automatically by following the algorithm listed in the pseudo code 6.4. It is interacting with the Event Log Mapping Database (Introduced in Section 4.2.6). All the queries are created without user interaction.

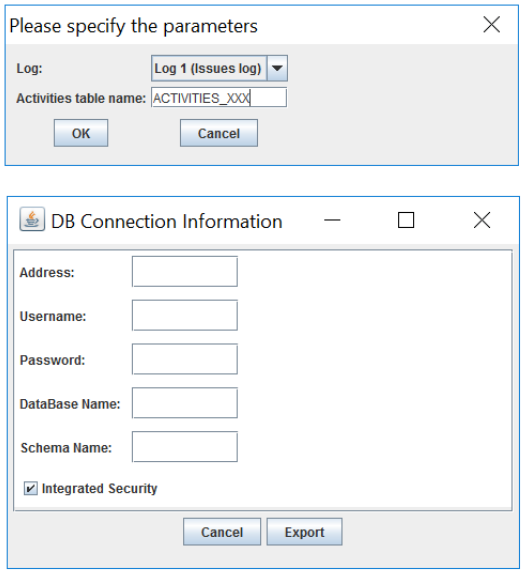

Figure 6.9: Activities table name and Extraction Database credentials.

```
def GenerateSimpleEventLog():

  var   dataBaseInstance    #Event Log Mapping Database Instance
  var   selectedLog         #Selected event log
  var   currentTrace        #Current trace
  var   currentEvent        #Current event
  var   currentCaseId       #Current case id
  var   currentEventName    #Current event name
  var   currentTimestamp    #Current timestamp
  var   eventLogRow         #Row consisting of caseId, eventName, timestamp
  var   simpleEventLog      #List of eventLogRow

  for currentTrace in selectedLog:
      currentCaseId = currentTrace.getCaseId()
      for currentEvent in currentTrace:
          currentEventName = currentEvent.getName()
          currentTimestamp = currentEvent.getTimestamp()
          eventLogRow.add(currentCaseId, currentEventName, currentTimestamp)
          simpleEventLog.add(eventLogRow)

  return simpleEventLog
```

Listing 6.4: Basic event log generation pseudo code.

The user can add additional attributes to the activities table by selecting them from a list obtained from the Event Log Mapping Database. This list contains all the attributes from all the events. Any attribute that is selected from this list will be added to the activities table as a new column. In the case the attribute is not included in a specific activity, a null value will be included. In the practical examples from Chapter 7, additional attributes should be added to the activities table. The final activities table should contain the resource that executed the task in one column. To produce this result, it is needed to *"Merge"* attributes, as this information is not contained

New Step	Original Step	Modification Stage	Solution for	ID
Data Preparation	-	Replication Database Stage	Ignore non-needed information.	B
			Activities discovery.	K
			Activities granularity discovery.	S
Manual DB Schema Identification*	Database Schema Identification	Replication Database Stage	Database schema identification	E
Artifact Manipulation*	Artifacts Schema Identification	Artifacts Stage	Removal of non-needed attributes.	N
			Removal of non-needed activities.	O
			Removal of repeated activities.	P
			Activity Naming.	Q
			Addition of static attributes.	R
			Activities grouping.	T
Export	-	Event Log Mapping Stage	Activities table generation.	U

Table 6.5: Summary of new steps in the Artifact-Centric Approach

in the same attribute for all the events. However, the merge is not that simple, because in some occasions both columns contain a different value and the selection of the right one depends on the current activity. Listing 6.5 provides a pseudo code of an algorithm to add selected attributes to the activities table.

```
def AddAttributesToEventLog():

 var dataBaseInstance        #Event Log Mapping Database Instance
 var selectedAttributesList  #List of selected attributes by the user.
 var selectedLog             #Selected event log
 var currentTrace            #Current trace
 var currentEvent            #Current event
 var currentCaseId           #Current case id
 var currentEventName        #Current event name
 var currentTimestamp        #Current timestamp
 var eventLogRow             #CaseId, eventName, timestamp and attributes
 var eventLog                #Final event log

 for currentTrace in selectedLog:
      currentCaseId = currentTrace.getCaseId()
      for currentEvent in currentTrace:
          currentEventName = currentEvent.getName()
          currentTimestamp = currentEvent.getTimestamp()
          eventLogRow.add(currentCaseId, currentEventName, currentTimestamp)

          if(selectedAttributeList>0):
              for attribute in selectedAttributeList:
                  eventLogRow.add(attribute)

          if(mergeColumns):
              merge(eventLog, columnToMerge1, columnToMerge2)

   return eventLog
```

Listing 6.5: Pseudo code to add attributes in the activities table.

The second objective of this phase is to export the activities table into the Extraction Database. The user needs to input the database credentials and the activities table name. If the information is submitted correctly, XTract V3 will transfer the generated activities table into the specified database. Both of these interfaces are displayed in figure 6.9.

In table 6.5 the new steps, original steps, modification stages and solved features are presented. The steps that contain a "*" are included into one of the original steps. Within this chapter we have evaluated XTract V1 and detected the missing features needed to obtain event logs for cloud systems. Moreover, we have presented the design and implementation algorithms included in XTract V3. The new steps and original steps in the Artifact-Centric Approach can be seen in

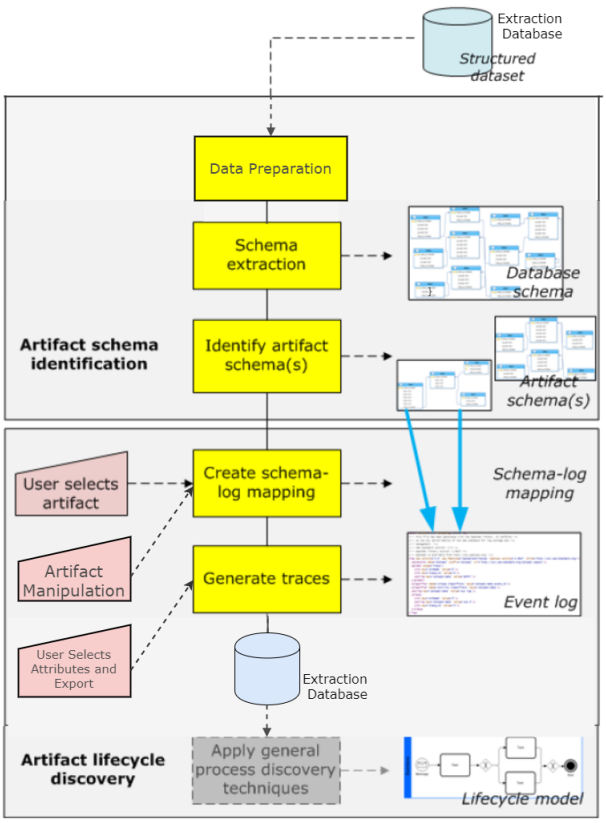

Figure 6.10: Artifact-Centric Approach New Steps.

figure 6.10. Finally, figure 6.11 and table 6.6 display the additions to the XTract V3 interface. These additions are highlighted in green.

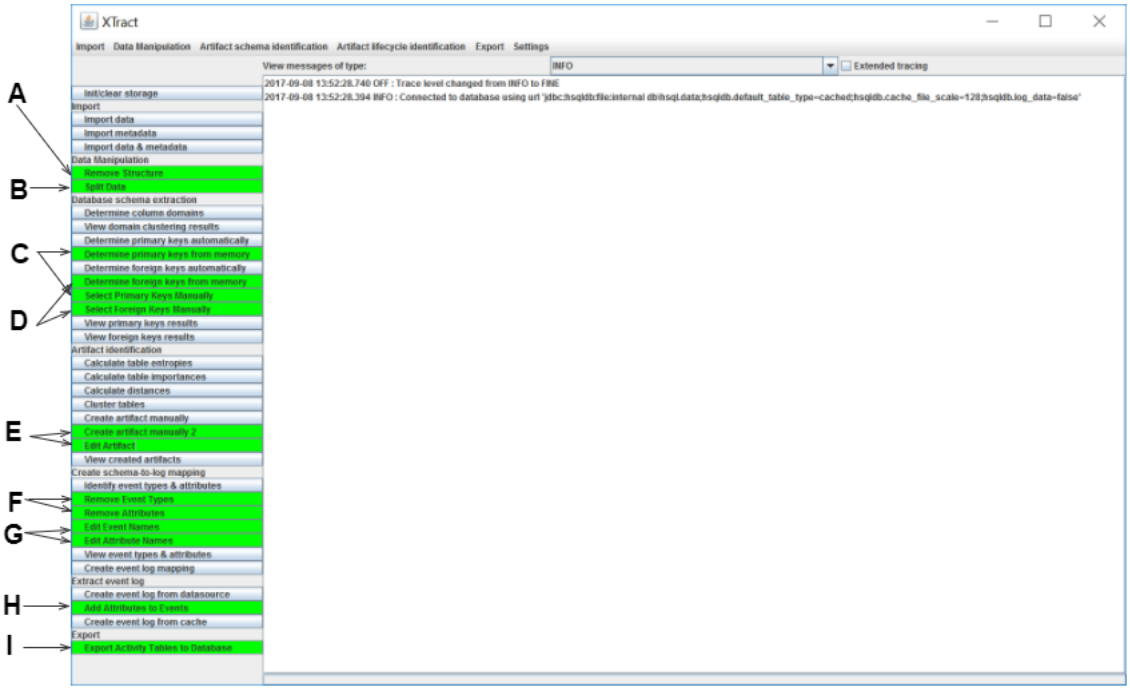

Figure 6.11: XTract V3 Additions.

ID	Feature
A	Remove Tables or Columns
B	Split Tables
C	Determine PKs manually or from memory
D	Determine FKs manually or from memory
E	Create and edit artifacts
F	Remove event types of attributes
G	Edit event type or attributes names
H	Add static attributes to events
I	Export activities table

Table 6.6: XTract V3 Additions from figure 6.11

Chapter 7

Transformation Phase Evaluation

The goal of this chapter is to present the evaluation objectives. The evaluation objectives are defined with the following questions 1) Is the API extractor successfully obtaining the needed information? 2) Does the Traditional Approach produce the expected results for cloud systems?, 3) Does the Artifact-Centric Approach work properly and how does the effort spent compare to the Traditional Approach?

To answer these questions, two completely independent cloud systems were selected. Section 7.1 introduces both systems and the target processes by showing the different objects and extracted relational databases. After the systems are introduced, the challenge of generating event logs will be solved with the Traditional Approach and the Artifact-Centric Approach. In Section 7.2 the Traditional Approach will be illustrated by displaying part of the SQL scripts and general guidelines on how to create them. In Section 7.3 a step by step guide of the decisions a user has to make when using XTract V3 will be shown. Finally, a comparison of the results obtained from both approaches will be produced in Section 7.4 and a discussion on the required amount of effort when using the Artifact-Centric Approach will be presented in Section 7.5.

7.1 Cloud Systems Structure and Extracted Information

This section introduces both cloud systems and the extracted relational databases obtained with the API Extractor. The aim of this section is to understand the size of the data set and validate that the extracted information is complete and accurate.

7.1.1 Jira

Jira is a proprietary issue tracking product developed by Atlassian Corporation. It provides bug tracking, issue tracking and project management functions. For the case study Jira Software will be the target system. The selected process is called Issue Management. In figure 7.1 the extracted relational database is shown. Details on the data schema are omitted in this public report.

The extracted information is complete in the sense that all the relevant information is included. The extracted information was reviewed by looking into Jira interface. A detailed review of specific issues was conducted to compare the information shown in the interface and the extracted information. In the first case study, the created relational database consists of 16 tables using 28 MB of storage.

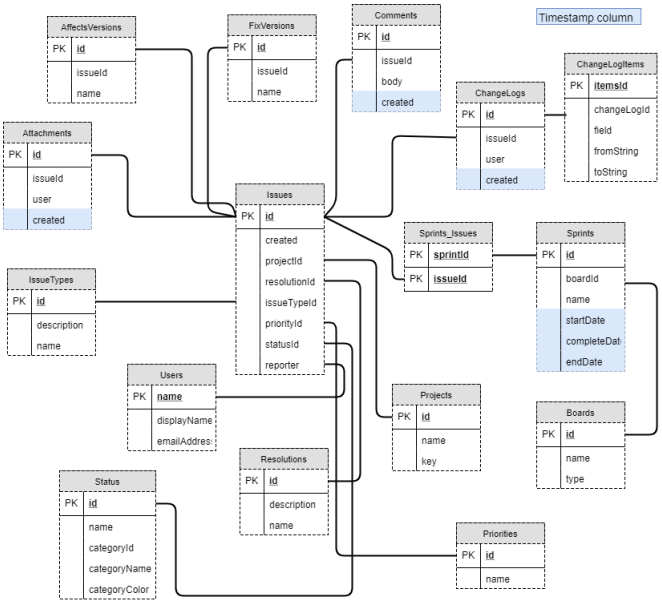

Figure 7.1: Jira extracted relational database, only relevant columns are shown.

7.1.2 Salesforce

Salesforce is a Customer Relationship Management product. There are two main offerings, Salesforce for Sales and Salesforce for Cloud. In figure 7.2 it is possible to see the extracted relational database and in table 7.2 the number of extracted rows per table.

The extracted relational database containing Salesforce information contains 10 tables and 11.50 MB of storage.

Table Name	Number of Records	Number of Columns
AffectsVersions	225	3
Attachments	448	5
Boards	12	3
ChangeLogItems	14709	8
ChangeLogs	11036	4
Comments	744	6
FixVersions	734	3
Issues	1459	16
IssueTypes	6	3
Priorities	6	2
Projects	3	5
Resolutions	8	3
Sprints	3	7
Sprints_Issues	71	2
Status	11	6
Users	68	3

Table 7.1: Amount of rows and columns per table for Jira.

Table Name	Number of Records	Number of Columns
Account	729	77
CaseComment	318	9
CaseContactRole	19	9
CaseFeed	2986	16
CaseHistory	3350	7
Cases	627	34
CaseStatus	13	10
Contact	1883	51
RecordType	20	13
Users	125	69

Table 7.2: Amount of rows and columns per table for Salesforce.

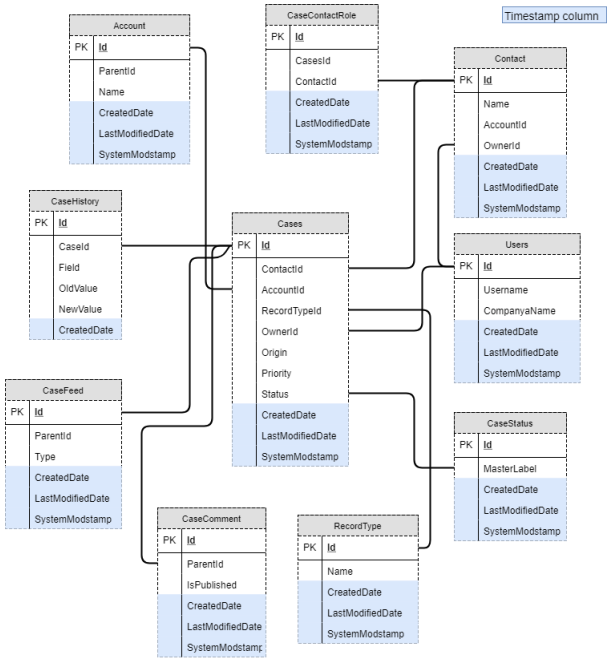

Figure 7.2: Salesforce extracted relational database, only relevant columns are shown.

7.2 Traditional Approach

So far we have validated that the extracted data is complete and accurate, therefore, the API Extractor fulfilled its purpose. Once the data is in the Extraction Database, the Transformation Phase begins. The next challenge is the generation of the event logs. The procedure defined in Chapter 2 was used in order to create the SQL scripts. For Jira, the event log contained 1,459 cases and the amount of events in all cases together is 9,279. For Salesforce, The final script produced 627 cases and 2863 total activities. Details of the extraction procedure and validation of results in the IT company are omitted from this public report.

7.3 Artifact-Centric Approach with XTract 3

In the following section we use the Artifact-Centric Approach to generate an event log from the extracted data. The features explained in Chapter 6 were implemented into XTract v3. In Section 7.4 a comparison between the Traditional Approach and Artifact-Centric Approach results will be discussed, evaluating if both approaches produced the same results and what is the amount of effort required in each of them. Specific choices made during the extraction from a given system are omitted from this public report.

Data Preparation

During the Data Preparation Phase, all the split operations introduced in Section 6.2.1 should be made. In figure 7.3 and 7.4, the interface to select the tables, *field*, *from* and *to* columns is shown. Besides both changes tables, the key relation between the tables should be selected.

After the values are selected correctly, the possible activities contained in the changes log will be shown. In this step, only the required activities should be chosen. Furthermore, the granularity level can be decided. Figure 7.5 illustrates the selection window. Once the tables are split, no further changes need to be applied in the Data Preparation Phase.

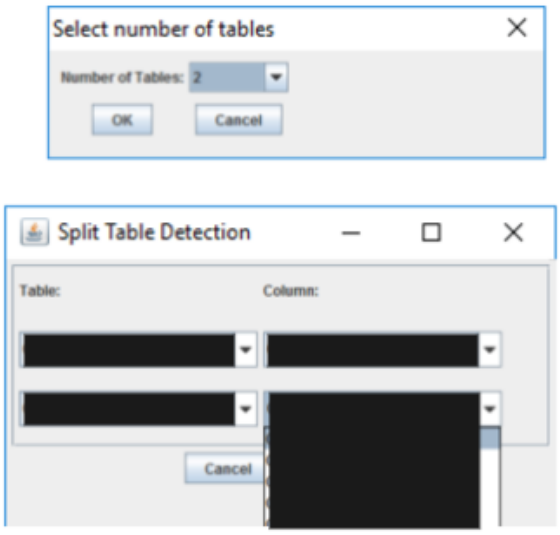

Figure 7.3: Selection of tables to split and their primary keys.

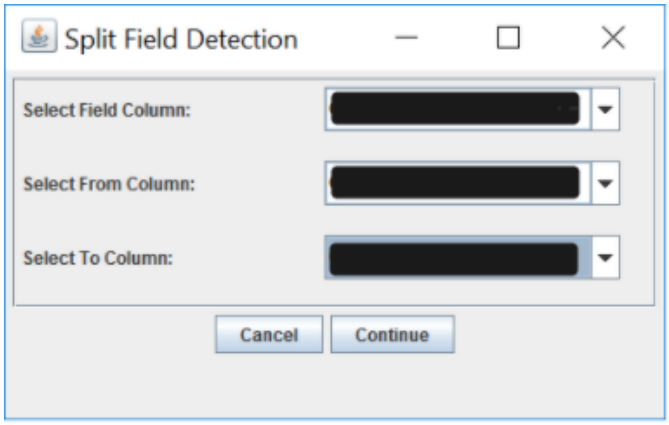

Figure 7.4: Selection of field, from and to columns.

Database Schema Identification

The first step is the domain identification, which was able to find all the value types. Once the domains are discovered, detection of primary and foreign keys are required. In image 7.6 manual selection of primary keys is displayed. In the case of foreign keys, automatic detection provides suggestions of foreign keys. Figure 7.7 shows the selection of foreign keys.

Artifact Schema Identification

The goal of this step is to combine all the tables into one artifact and use Issues as the main table. Here, both manual specification of artifacts or automatic discovery through clustering of tables can be used. The artifact and its tables can be seen in 7.8.

Artifact Manipulation

Once the artifact, its main table and secondary tables are discovered, all the attributes and event types can be identified. After these elements are identified, they can be removed. In this

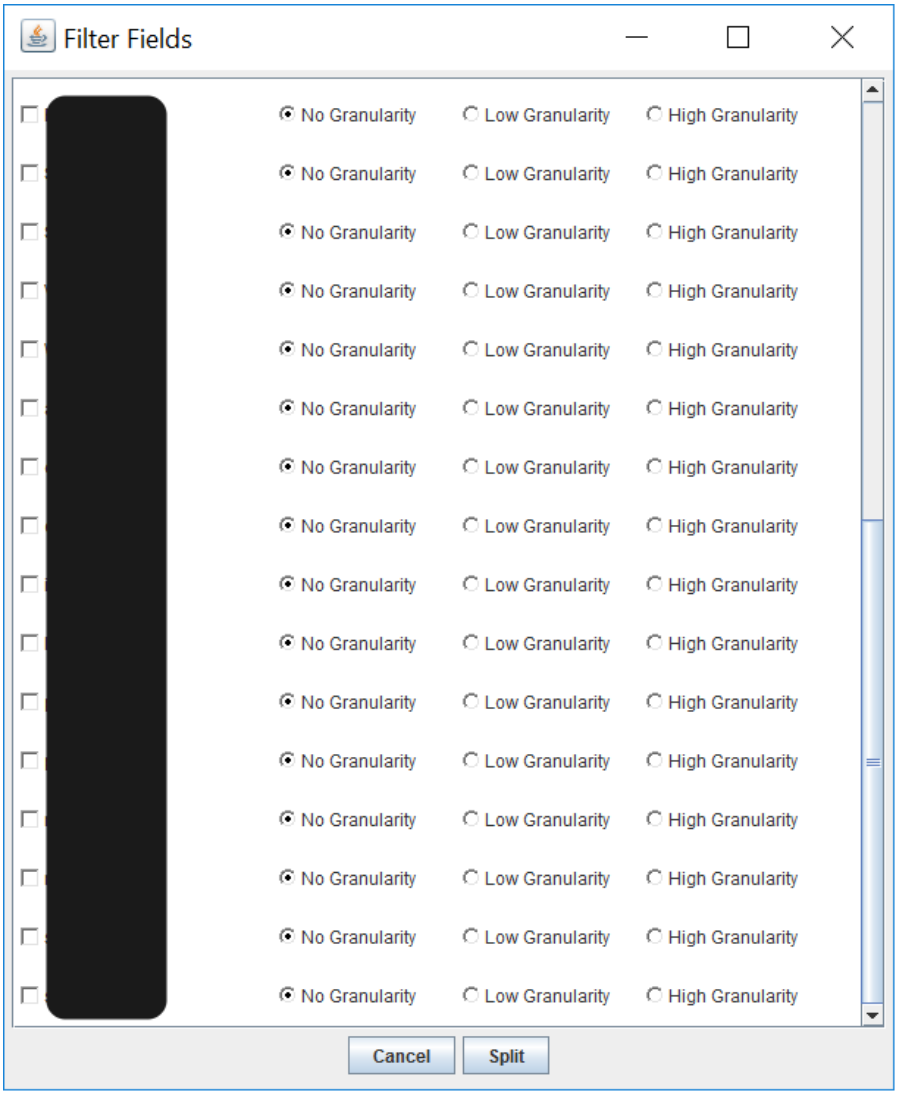

Figure 7.5: Granularity level selection.

case, there are several event types that were removed.

Additionally, activities can be renamed in this step. Activities that contain a "S" in their table name were generated from changes tables. The default names take into account the tables and columns names only. The naming window is shown in figure 7.10. In this case, a static attribute for sorting is required. This is illustrated in figure 7.9.

Schema to Log Mapping and Trace Generation

The Schema to Log Mapping and Trace generation is very straightforward as there are no important decisions to make. The end result of this step is a XES file containing all the traces.

Export

During this step a table containing the basic event log is generated. Section 6.2.4 discussed and presented how to include additional attributes into the event log. The algorithm presented in that section is used in this step of the procedure. Additionally to the resources, the sorting attribute needs to be chosen too. Figure 7.11 shows the selection interface.

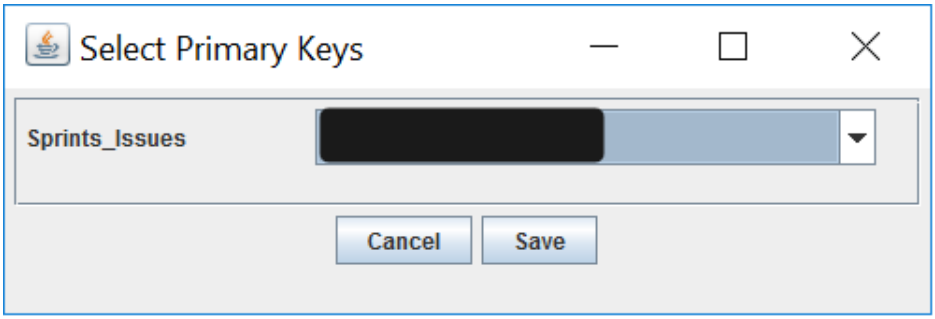

Figure 7.6: Manual selection of primary key.

The activities table can be transferred to any database. As it was explained in our Technical Setup (Section 4.2), this table should be located in the Extraction Database. The export functionality requires the target database credentials in order to transfer the activities table.

The Artifact-Centric Approach yielded the same results as the Traditional Approach for both systems. In the case of Jira, 1,459 cases and 9,279 total activities were found, while in Salesforce 627 cases and 2,863 total events. This concludes the Transformation Phase. The next step is to evaluate both approaches from two perspectives. Section 7.4.1 makes a more detailed comparison between both event logs. Section 7.4.2 takes a look into the amount of user effort needed in each technique.

True foreign key	Reference name	Parent table name	Parent column name	Child table name
☑	Candidate_Issues_	Issues	id	Affe
☑	Candidate_Issues_	Issues	id	S
☑	Candidate_Issues_	Issues	id	S
☑	Candidate_Issues_	Issues	id	S
☑	Candidate_Issues_	Issues	id	S
☑	Candidate_Issues_	Issues	id	S
☑	Candidate_Issues_	Issues	id	S
☑	Candidate_Issues_	Issues	id	S
☑	Candidate_Issues_	Issues	id	S
☑	Candidate_Issues_	Issues	id	S
☑	Candidate_Issues_	Issues	id	S
☑	Candidate_Issues_	Issues	id	S
☑	Candidate_Issues_	Issues	id	S
☑	Candidate_Issues_	Issues	id	S
☑	Candidate_Issues_	Issues	id	S
☑	Candidate_Issues_	Issues	id	S
☑	Candidate_Issues_	Issues	id	S_

Figure 7.7: Foreign Keys Jira.

Artifact id	Artifact name	Table name	Is main table
1	Issues	Affe	☐
1	Issues	A	☐
1	Issues	B	☐
1	Issues	C	☐
1	Issues	F	☐
1	Issues	I	☐
1	Issues	I	☑
1	Issues	F	☐
1	Issues	F	☐
1	Issues	F	☐
1	Issues	S	☐
1	Issues	S	☐
1	Issues	S	☐
1	Issues	S	☐
1	Issues	S	☐
1	Issues	S	☐
1	Issues	S	☐
1	Issues	S	☐
1	Issues	S	☐
1	Issues	S_	☐

Figure 7.8: Artifact discovery results.

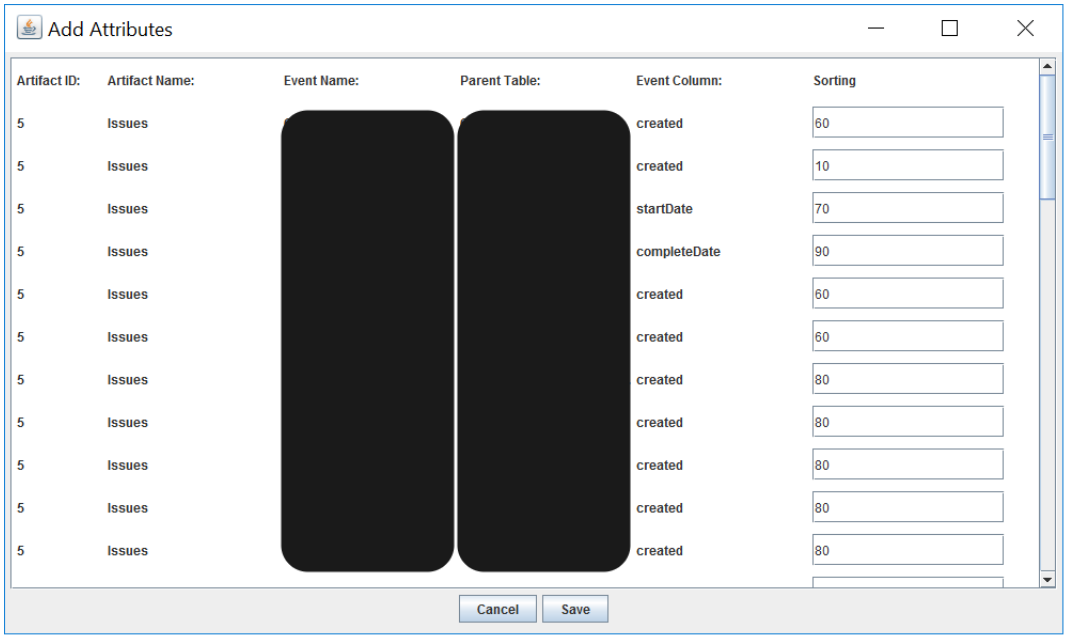

Figure 7.9: Adding Sorting column for each activity.

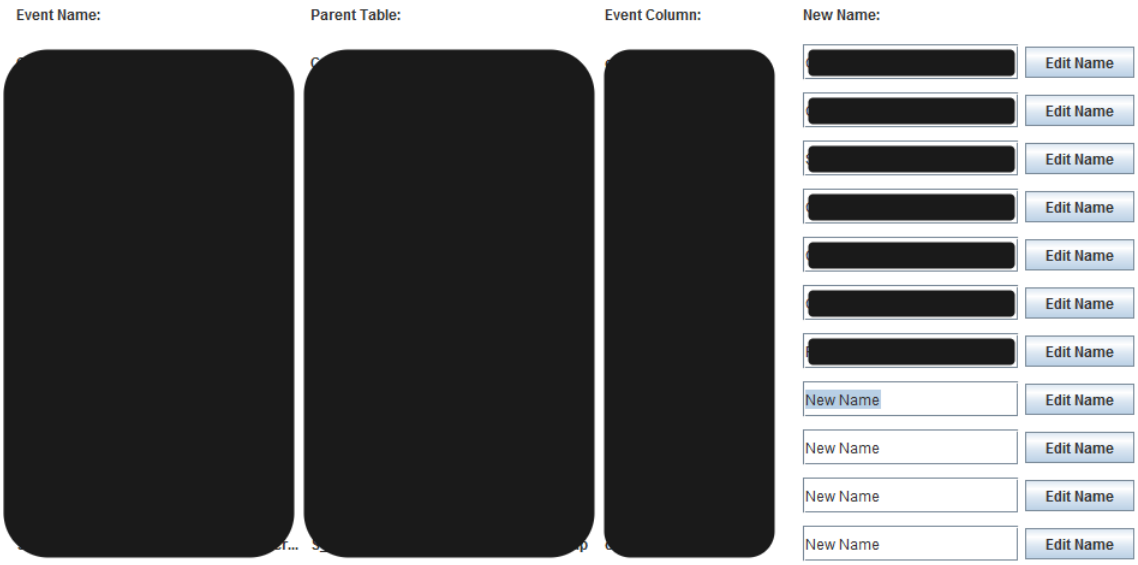

Figure 7.10: Editing event names.

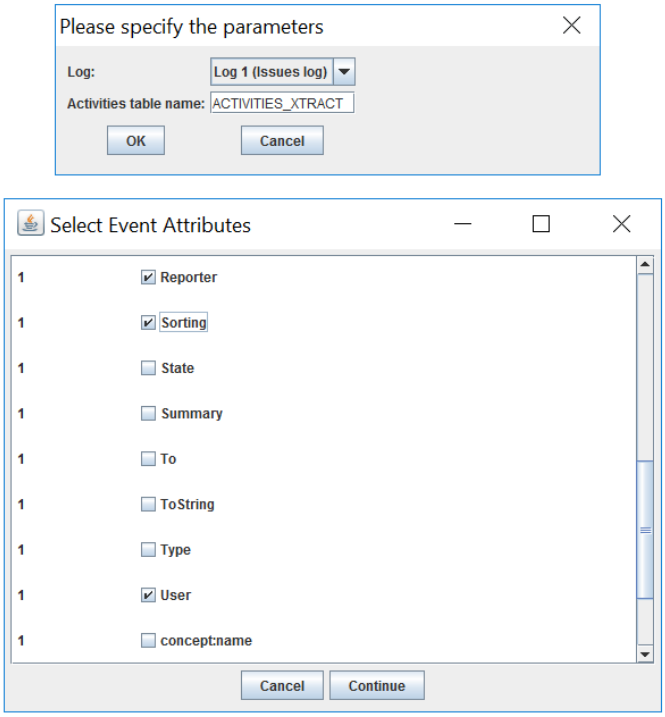

Figure 7.11: Selecting additional attributes.

7.4 Evaluation of Results

The event log evaluation will be divided in two parts. The first type of evaluation will compare basic event logs. A basic event log consists of a case id, an activity name and a time stamp. The second part takes into consideration the sorting and resources column. At the end, a summary of the non-automatic actions will be displayed. This summary aims to ponder on which steps are completely automatic, which steps require selection of parameters only and which steps require domain knowledge. In Chapter 10, a discussion on how to make the approach more automatic will be discussed.

7.4.1 First Level Results - Basic Event Log

As it was mentioned in Section 7.3, the Artifact-Centric Approach obtained the same results as the Traditional Approach. When evaluating the basic event log, the only difference is in the order of activities. The amount of cases and activities per case is the same. The order is different in the traces where activities have the same time stamp. This problem is solved by using a sorting column to indicate which activity should be displayed first. Tables 7.3 and 7.4 show one trace obtained with Traditional Approach and the Artifact-Centric Approach respectively. The difference between the order of the event logs is highlighted.

7.4.2 Second Level Results - Complete Event Log

The second level results compare the results once the sorting column and the resources are included. After both columns are included, both event logs are completely equal. Figures 7.12 and 7.13 show the results from the Traditional Approach and the Artifact-Centric Approach respectively.

Traditional Approach		
CASE_ID	**ACTIVITY_NAME**	**EVENT_TIME**
14401	Create Issue	2016-04-15 10:25:37.000
14401	Change Status to In evaluation	2016-04-20 20:12:38.000
14401	Change Status to Backlog	2016-08-05 18:40:28.000
14401	Change Issue Type	2016-08-05 18:43:46.000
14401	Change Status to In evaluation	2016-08-05 18:44:28.000
14401	Change Status to Backlog	2016-08-09 22:27:58.000
14401	Change Status to Hold	2016-08-09 22:28:51.000
14401	***Change Status to Declined***	***2016-08-25 20:26:59.000***
14401	***Change Content***	***2016-08-25 20:26:59.000***
14401	Change Issue Type	2016-11-15 15:49:16.000

Table 7.3: Obtained result with the Traditional Approach (Simple Event Log).

Artifact-Centric Approach		
CASE_ID	**ACTIVITY_NAME**	**EVENT_TIME**
14401	Create Issue	2016-04-15 10:25:37.000
14401	Change Status to In evaluation	2016-04-20 20:12:38.000
14401	Change Status to Backlog	2016-08-05 18:40:28.000
14401	Change Issue Type	2016-08-05 18:43:46.000
14401	Change Status to In evaluation	2016-08-05 18:44:28.000
14401	Change Status to Backlog	2016-08-09 22:27:58.000
14401	Change Status to Hold	2016-08-09 22:28:51.000
14401	***Change Content***	***2016-08-25 20:26:59.000***
14401	***Change Status to Declined***	***2016-08-25 20:26:59.000***
14401	Change Issue Type	2016-11-15 15:49:16.000

Table 7.4: Obtained result with the Artifact Centric Approach (Simple Event Log).

7.5 How much effort is needed?

CASE_ID	ACTIVITY_NAME	EVENT_TIME	userName	sorting
14672	Create Issue	2016-05-10 14:22:00.000	v[redacted]	10
14672	Change Status to In evaluation	2016-05-11 19:48:20.000	m[redacted]	20
14672	Change Content	2016-08-05 15:05:25.000	m[redacted]	60
14672	Change Status to Backlog	2016-08-05 18:15:52.000	m[redacted]	20
14672	Change Status to In evaluation	2016-08-05 18:41:30.000	m[redacted]	20
14672	Change Issue Type	2016-08-12 15:55:07.000	a[redacted]	50
14672	Change Status to Backlog	2016-08-12 15:55:12.000	a[redacted]	20
14672	Change Status to Hold	2016-08-12 15:55:14.000	a[redacted]	20
14672	Change Status to Declined	2016-08-19 18:39:05.000	m[redacted]	20
14672	Change Content	2016-08-19 18:39:05.000	m[redacted]	60
14672	Change Issue Type	2016-11-15 15:49:16.000	m[redacted]	50

Figure 7.12: Traditional Approach Results.

To understand the needed effort we will review the manual work at each step of the Artifact-Centric Approach. Table 7.5 summarizes all the steps and the amount of non-automatic actions. It also illustrates the actions which require domain knowledge. In this public version of the report, several details on the comparison to existing techniques are omitted.

The *Data Preparation* step requires several non-automatic actions. A number of split operations need to be exercised in the Replication Database. These split operations depend on each system, the specific process and the target event log. They can be divided into I) Split changes tables and II) Split objects tables. Both changes table representation options need to be considered. The reason for splitting changes tables was introduced in Section 6.2.1 and changes tables representation options were explained in Section 4.4.1. After splitting the tables, the user needs to select which information coming from the field column in is relevant and its granularity level. Although all these operations are based in parameters, the selection to split non-changes tables, identification of the desired new tables and the granularity level require domain knowledge.

When identifying the *database schema* in the case studies, several semi-automatic steps were needed. In the case of Jira, primary keys were not completely discovered. For foreign keys, XTract V3 suggested some of the relations, but the users need to validate the results and add missing

CASE_ID	ACTIVITY_NAME	EVENT_TIME	User	Sorting
14672	Create Issue	2016-05-10 14:22:00.000	v.z[redacted]	10
14672	Change Status to In evaluation	2016-05-11 19:48:20.000	m.[redacted]	20
14672	Change Content	2016-08-05 15:05:25.000	m.[redacted]	60
14672	Change Status to Backlog	2016-08-05 18:15:52.000	m.[redacted]	20
14672	Change Status to In evaluation	2016-08-05 18:41:30.000	m.[redacted]	20
14672	Change Issue Type	2016-08-12 15:55:07.000	a.l[redacted]	50
14672	Change Status to Backlog	2016-08-12 15:55:12.000	a.l[redacted]	20
14672	Change Status to Hold	2016-08-12 15:55:14.000	a.l[redacted]	20
14672	Change Status to Declined	2016-08-19 18:39:05.000	m.[redacted]	20
14672	Change Content	2016-08-19 18:39:05.000	m.[redacted]	60
14672	Change Issue Type	2016-11-15 15:49:16.000	m.[redacted]	50

Figure 7.13: Artifact-Centric Approach Results.

Artifact-Centric Step	Non-Automatic Actions	Domain knowledge?
Data Preparation	Split Operations	Yes
Database Schema Identification	Manual identification of PK, FK.	Yes
Artifact Schema Discovery	Main Table selection	Yes
Artifact Manipulation	Removal of non-needed activities	Yes
	Activity naming	Yes
	Addition of static attributes	Yes
Export	Selection of additional attributes	Yes

Table 7.5: Non-automatic actions summary.

foreign keys. Nevertheless, this step is only needed when the information of the database schema is not known and it needs to be completed one time only. When the database schema is known, it can be automatically loaded into XTract v3, making this step completely automatic.

As it is possible to see in table 7.5, *artifact schema discovery* and *artifact manipulation* together have four non-automatic actions. Main table selection depends on the target event log. It is possible that the clustering approach is not able to detect the right main table. In that scenario, a manual change of main table needs to be executed. Removal of non-needed activities, activity naming and addition of static attributes are actions that are strictly related to domain knowledge.

Finally, the *export* step can be divided into three parts. The first part is completely automatic and is not included in table 7.5. This part is the creation of an activity table with case id, activity names and time stamp. The second part, which is related to the addition of extra columns such as sorting and resources, requires domain knowledge. The user needs to specify which additional columns should be added. The third part needs the information about the Extraction Database in order to transfer the activities table. In conclusion, only the second part needs domain knowledge.

The key difference between the Traditional Approach and the Artifact-Centric Approach is that the former is based on SQL queries. This means that a person with extended SQL knowledge needs to create the scripts. The main advantage of the Artifact-Centric Approach is that the SQL queries are produced automatically and ran internally. Consequently, a user without extended SQL knowledge can use XTract v3 successfully.

The schema detection phase can require a lot of effort. When the schema is not available, the user will need to create it. This effort is directly related to the number of tables, primary keys

and foreign key relations.

In this section all the non-automatic steps for the Artifact-Centric Approach were presented. Most of them only require certain parameters. Furthermore, the differences between the Traditional Approach and the Artifact-Centric Approach were summarized. The most important advantage of the Artifact-Centric Approach being the automatic generation of SQL queries. Different ideas on how to reduce the amount of effort from the Artifact-Centric Approach will be presented in the Future Work section in Chapter 10.

Chapter 8

Process Mining Analysis

The goal of this chapter is to introduce the procedure followed in the Analysis Phase to obtain process insights. The followed procedure consists of reading the event log with a process mining tool in order to generate a model. Once a model is discovered, it is possible to study individual cases by applying multiple filters. Section 8.1 covers the refinement of the needed features presented in Chapter 4. Section 8.2 discusses the created key performance indicators and introduces the developed analysis. Section 8.3 explains the data set constrains. Finally, section 8.4 displays some of the findings. The analysis discussed in this section have been implemented as closed-source projects in the IT company. Several details of the analysis are omitted from this public report.

8.1 Refinement of needed features in the Analysis Phase

The following elements are the needed features for the Analysis Phase established in Section 4.4.3 (Table 4.6).

S) Activity grouping

This feature was covered in the Transformation Phase.

X) Model discovery

Model discovery is handled by the process mining tool.

W) Process Mining Analysis

As it was mentioned in Chapter 2, Process Mining Analysis was made using a process mining tool.

Analysis Name	System
Open Issues	Jira
Resolution Time	Jira
Open Cases	Salesforce
Resolution Time	Salesforce

Table 8.1: Developed analyses.

8.2 Analysis

Table 8.1 displays the four created analyses. Different set of key performance indicators were created. In the case of *Open Issues* and *Open Cases*, *Working Time* and *Life Time* were created. For Resolution Time, the metric *Resolution Time* was defined. Figure 8.1 and 8.2 show these metrics in a horizontal time line representation.

Working Time: Amount of time since an employee started to work on an issue or case. This metric starts once the status of the issue or case is *In progress*.

Resolution Time: Amount of time since an issue or case was created until the resolved date.

Life Time: Amount of time since an issue or case was created until the present date or the resolved date. This formula was created for the Open Issues analysis.

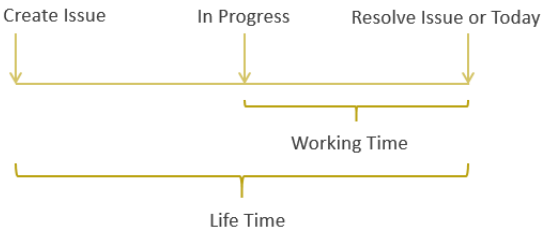

Figure 8.1: Open Issues key performance indicators.

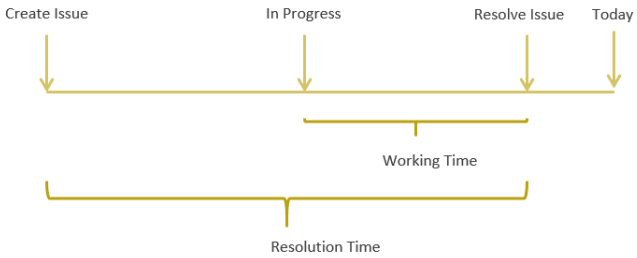

Figure 8.2: Resolution Time key performance indicators.

8.3 Data Sets Constraints

There have been different publications analyzing process mining for similar systems. For instance, M. Gupta et al [30] looked into three different software repositories systems used in the Google Chromium project. 9,744 traces were analyzed in that case. In [23] a study on how to reduce user input requests in tickets life cycle was produced by looking into 593,497 closed cases by M. Gupta et al. Ortu et al analyzed more than 700K cases from more than 1K projects from Jira repositories [36]. In [38] process mining techniques were applied into a repository consisting of 42,373 bugs, 495,321 comments and 13,944 users by Poncin et al. The present work uses a much-limited amount of data. In the case of Jira, the data set contains 1,459 traces, while the Salesforce case contains 627 traces. Consequently, it is not possible to obtain general findings such as the ones presented in the mentioned publications. Nevertheless, very specific findings for the Resolution Time app will be presented in the next section.

8.4 Findings

In the case of Salesforce Resolution Time, the metric *Resolution Time* is 20.2 days, while the *Working Time* is 23.7 days. The reason for *Working Time* taking more days than *Resolution Time* is because some cases are resolved before the case is put into the In Progress status. In those cases, the *Working Time* will not be counted on the calculations, causing the *Resolution Time* average to be less than the *Working Time*. Further, we could identify outlier cases with significantly higher resolution time. By filtering the extracted event log for these cases specifically and visualizing them in a process mining tool, we obtained process models as illustrated in Figure 8.3. From these models, one can see that delays occurred after changes in priority or after change of assignee.

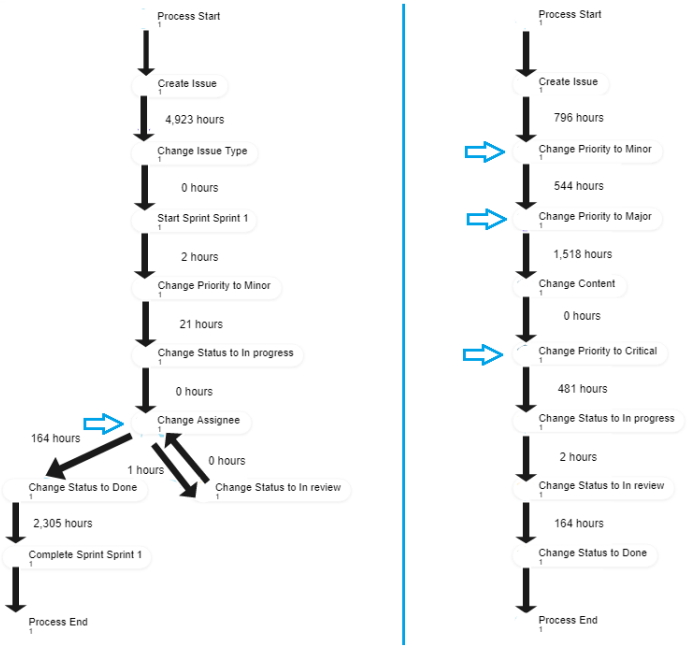

Figure 8.3: Delays caused by change of priority or change of assignee.

Chapter 9

Process Dimensions

The aim of this chapter is to demonstrate another advantage of the Artifact-Centric Approach. This advantage is the possibility to change dimensions to analyze processes from different angles. Several studies have explained the benefits of changing dimensions in an event log using process cubes [7], [8], [51], [44], [28]. In Section 9.1 an introduction to the concept of changing dimensions is introduced. In Section 9.2 the interface to manipulate an artifact with XTract V3 is shown. Finally, in Section 9.3, the benefits of changing dimensions using the Artifact-Centric Approach are summarized.

9.1 Process Dimensions

In order to understand the concept of process dimensions, process cubes have been proposed in the past. A process cube is a multidimensional structure built from event log data in a way that facilitates further meaningful process mining analysis [28]. The notion of changing dimensions with the Artifact-Centric Approach is the possibility to produce event logs related to the same process by changing the case id from the created artifacts.

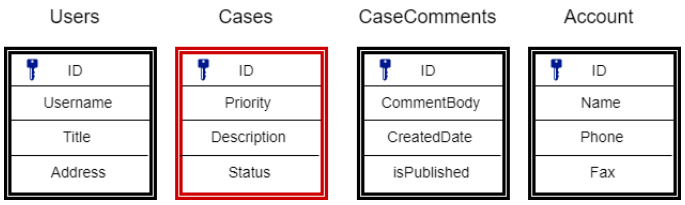

Figure 9.1: Case Id Selection.

In the context of automatic event log generation, there are two factors to take into account when changing dimensions. It is possible to create one dimension and multiple dimension event logs. A dimension can be defined as the case id of an event log. An artifact is a group of tables that contain all the information from a process. This group consists of a main table and secondary tables. The primary key from the main table is the case id, while the whole artifact is considered a process. One dimensional event logs follow this definition. In the case of multiple dimension event logs, the concept changes. The main table of an artifact should be the combination of two or more tables. By combining primary keys, the case id is represented by multiple dimensions. In the present work, we look at the first option and the second option is included in the future work section in Chapter 10.

In figure 9.1, we provide some dimension possibilities from the second case study. In the case study we selected the table *Cases* as the dimension we would like to analyze. Nevertheless, it is also possible to pick any other table as the case id.

9.2 Changing Dimensions with XTract V3

In order to change a dimension with XTract V3, the edit artifact feature can be used. The desired artifact and the new main table should be selected. In figure 9.2, it is possible to see the graphical interface. After the artifact is discovered, the steps to generate the activities table are the same as in the case studies. Detailed findings obtained this way are omitted from the public version of this report.

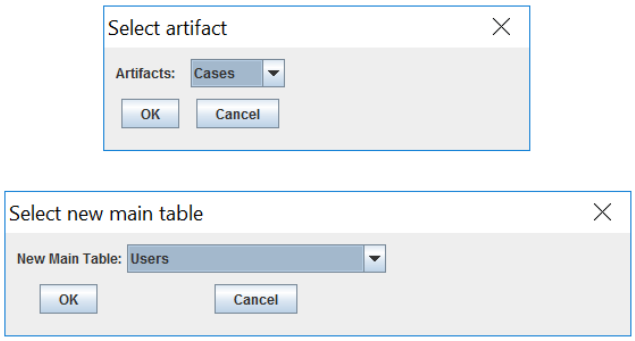

Figure 9.2: Graphical interface to edit an artifact.

9.3 Advantages

The most important benefit regarding process dimensions when using the Artifact-Centric Approach and XTract V3 is the saved amount of time. By changing the main table, it is possible to generate a different activity table in a short amount of time. Different activities tables allow for multiple visualizations options. There is no limitation in the case id selection.

Chapter 10

Conclusion

During this Master thesis we conducted extensive research in order to produce the complete process mining cycle for cloud systems. Emphasis was placed into solving the challenge of data transformation to generate event logs. Many aspects were taken into account to solve this challenge. Information systems, traditional data transformation procedures, different transformation approaches, and different software components. All these factors produced the following research problem:

> *Given a **cloud system**, improve the transformation step of the raw data into event logs by providing a methodology, which can A) reduce the expertise required to transform the data, B) produce the same results as the current traditional approach used by the engineers C) be flexible enough to allow for domain knowledge decisions.*

In Section 10.1, we enumerate the developed contributions and how we have successfully created a tool, which reduces the required SQL knowledge while producing the same results as the traditional approaches. Section 10.2 provides three possible upgrades on XTract V3 that were not covered in this work.

10.1 Summary of Contributions

To solve the research problem, we created several technical components. First, we designed and developed an API extractor to obtain the information in the Extraction Phase. This tool was created for Jira Software and we re-used another API extractor for Salesforce. For the Transformation Phase, we developed two SQL scripts that produced event logs using the traditional approach. The API extractor and the SQL scripts were developed as closed-source components.

The third and most important contribution is the creation of a prototype (XTract V3) that reproduces the theory behind the Artifact-Centric Approach. To do so, it integrates new steps into the original Artifact-Centric Approach. These steps are 1) Data Preparation, 2) Artifact Manipulation and 3) Export. These new steps are able to solve the *"Vertical Anti-Partitioning"* characteristic while also defining granularity levels on activities. By using XTract V3, non-technical users without SQL expertise can generate event logs as the software handles the queries internally. Moreover, it enables a fast and reliable way to produce different event logs, opening the door for new process mining insights. XTract V3 was developed as an open-source tool [1].

[1] https://svn.win.tue.nl/repos/prom/XTract/

As last contribution, a process mining analysis or the Issue Resolution and the Case Management process were provided.

10.2 Future Work

Future work related to the IT company is omitted from this public report.

How Can It be More Automatic?

In XTract V3, vertical anti-partitioning was solved by taking into account user input about the tables and columns to split. However, there are some non-tested ideas to improve this method. The first idea consists of developing metrics, which determine how likely is a column holding different entities. There are some conditions that can help detect these columns. For instance, in the case studies, the *field* column never contained null values. Secondly, name detection on the tables and column names could be another solution. Since an API extractor needs to be developed for cloud systems, the names of tables and column can be updated in the Extraction Phase. Then, XTract could have name detection capabilities integrated to automatically detect changes tables. The disadvantage of this approach is that it transfers the manual overhead to the API extractor.

In the current version, the primary keys and foreign keys of the split tables need to be specified by the user after the Data Preparation step. However, this step could be removed by replicating the primary keys and foreign keys of the original tables.

Multi-Dimensional Case Ids

The ability to combine case ids could open the door for new kind of event logs. By combining primary keys of different tables, XTract v3 will allow multi-dimensional event logs.

Bibliography

[1] Ziawasch Abedjan and Felix Naumann. Advancing the discovery of unique column combinations. In *CIKM*, 2011. 25

[2] Arya Adriansyah, Natalia Sidorova, and Boudewijn F. van Dongen. Cost-based fitness in conformance checking. In *ACSD*, 2011. 7

[3] Arya Adriansyah, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Towards robust conformance checking. In *Business Process Management Workshops*, 2010. 7

[4] Arya Adriansyah, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Conformance checking using cost-based fitness analysis. *2011 IEEE 15th International Enterprise Distributed Object Computing Conference*, pages 55–64, 2011. 7

[5] Rakesh Agrawal, Dimitrios Gunopulos, and Frank Leymann. Mining process models from workflow logs. In *EDBT*, 1998. 7

[6] Robin Bergenthum and year=2007 Jörg Desel and Robert Lorenz and Sebastian Mauser, booktitle=BPM. Process mining based on regions of languages. 7

[7] Alfredo Bolt, Massimiliano de Leoni, Wil M. P. van der Aalst, and Pierre Gorissen. Exploiting process cubes, analytic workflows and process mining for business process reporting: A case study in education. In *SIMPDA*, 2015. 67

[8] Alfredo Bolt and Wil M. P. van der Aalst. Multidimensional process mining using process cubes. In *BMMDS/EMMSAD*, 2015. 67

[9] J.C.A.M. Buijs. Mapping Data Sources to XES in a Generic Way. Master's thesis, Eindhoven University of Technology, 2010. 8, 10, 11, 13

[10] Diego Calvanese, Marco Montali, Alifah Syamsiyah, and Wil M. P. van der Aalst. Ontology-driven extraction of event logs from relational databases. In *Business Process Management Workshops*, 2015. 8, 12

[11] Josep Carmona and Jordi Cortadella. Process mining meets abstract interpretation. In *ECML/PKDD*, 2010. 7

[12] Josep Carmona, Jordi Cortadella, and Michael Kishinevsky. A region-based algorithm for discovering petri nets from event logs. In *BPM*, 2008. 7

[13] year=2014 Christian W. Günther and Eric Verbeek. Where innovation starts xes standard definition. 10

[14] David Cohn and Richard Hull. Business artifacts: A data-centric approach to modeling business operations and processes. *IEEE Data Eng. Bull.*, 32:3–9, 2009. 9

[15] Jonathan E. Cook and Alexander L. Wolf. Discovering models of software processes from event-based data. *ACM Trans. Softw. Eng. Methodol.*, 7:215–249, 1998. 7

[16] Jonathan E. Cook and Alexander L. Wolf. Software process validation: Quantitatively measuring the correspondence of a process to a model. *ACM Trans. Softw. Eng. Methodol.*, 8:147–176, 1999. 7

[17] Ana Karla A. de Medeiros, A. J. M. M. Weijters, and Wil M. P. van der Aalst. Genetic process mining: an experimental evaluation. *Data Min. Knowl. Discov.*, 14:245–304, 2007. 7

[18] Eduardo González López de Murillas, Hajo A. Reijers, and Wil M. P. van der Aalst. Connecting databases with process mining: A meta model and toolset. In *BMMDS/EMMSAD*, 2016. 12

[19] Eduardo González López de Murillas, Wil M. P. van der Aalst, and Hajo A. Reijers. Process mining on databases: Unearthing historical data from redo logs. In *BPM*, 2015. 2, 8

[20] Schahram Dustdar and Robert Gombotz. Discovering web service workflows using web services interaction mining. *IJBPIM*, 1:256–266, 2006. 12

[21] Walid Gaaloul, Khaled Gaaloul, Sami Bhiri, Armin Haller, and Manfred Hauswirth. Log-based transactional workflow mining. *Distributed and Parallel Databases*, 25:193–240, 2009. 7

[22] Stijn Goedertier, David Martens, Jan Vanthienen, and Bart Baesens. Robust process discovery with artificial negative events. *Journal of Machine Learning Research*, 10:1305–1340, 2009. 7

[23] Monika Gupta, Allahbaksh Asadullah, Srinivas Padmanabhuni, and Alexander Serebrenik. Reducing user input requests to improve it support ticket resolution process. 2017. 64

[24] Emna Hachicha, Walid Gaaloul, and Zakaria Maamar. Social-based semantic framework for cloud resource management in business processes. *2016 IEEE International Conference on Services Computing (SCC)*, pages 443–450, 2016. 7

[25] Mieke Julie Jans. From relational databases to valuable event logs for process mining purposes: A procedure. 2017. 2, 8

[26] Xixi Lu. Artifact-Centric Log Extraction and Process Discovery. Master's thesis, Eindhoven University of Technology, 2013. iii, 2, 7, 11, 13, 25, 26, 27, 35, 36, 42

[27] Xixi Lu, Marijn Nagelkerke, Dennis van de Wiel, and Dirk Fahland. Discovering interacting artifacts from erp systems. *IEEE Transactions on Services Computing*, 8:861–873, 2015. iii, 8, 13, 26, 35

[28] T. Mamaliga. Realizing a Process Cube Allowing for the Comparison of Event Data. Master's thesis, Eindhoven University of Technology, 2013. 67

[29] Peter Mell and Tim Grance. The nist definition of cloud computing. 2010. 12

[30] Ashish Sureka Monika Gupta and Srinivas Padmanabhuni. Process Mining Multiple Repositories for Software Defect Resolution from Control and Organizational Perspective. pages 122–131, 2014. 64

[31] Jorge Munoz-Gama and Josep Carmona. A fresh look at precision in process conformance. In *BPM*, 2010. 7

[32] Brad A. Myers and Jeffrey Stylos. Improving api usability. *Commun. ACM*, 59:62–69, 2016. 12

[33] Anil Nigam and Nathan S. Caswell. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42:428–445, 2003. 9

[34] Erik H. J. Nooijen. Artifact-Centric Process Analysis. Master's thesis, Eindhoven University of Technology, 2012. iii, 2, 7, 9, 22, 35

[35] Erik H. J. Nooijen, Boudewijn F. van Dongen, and Dirk Fahland. Automatic discovery of data-centric and artifact-centric processes. In *Business Process Management Workshops*, 2012. iii, 8, 9

[36] Marco Ortu, Giuseppe Destefanis, Bram Adams, Alessandro Murgia, Michele Marchesi, and Roberto Tonelli. The jira repository dataset: Understanding social aspects of software development. In *PROMISE*, 2015. 64

[37] D.A.M. Piessens. Event Log Extraction from SAP ECC 6.0. Master's thesis, Eindhoven University of Technology, 2011. 2, 8

[38] Wouter Poncin, Alexander Serebrenik, and Mark van den Brand. Process mining software repositories. *2011 15th European Conference on Software Maintenance and Reengineering*, pages 5–14, 2011. 64

[39] A. Ramesh. Process mining in PeopleSoft. Master's thesis, Eindhoven University of Technology, 2006. 2, 8

[40] Martin P. Robillard and Robert DeLine. A field study of api learning obstacles. *Empirical Software Engineering*, 16:703–732, 2011. 12

[41] Andreas Sonntag and Peter Fettke. Efficiency of generated performer networks in collaborative business process models. *2016 IEEE 18th Conference on Business Informatics (CBI)*, 01:26–34, 2016. 7

[42] Wil M. P. van der Aalst. Configurable services in the cloud: Supporting variability while enabling cross-organizational process mining. In *OTM Conferences*, 2010. 2, 12

[43] Wil M. P. van der Aalst. Process mining - discovery, conformance and enhancement of business processes. *Journal of Biomedical Informatics*, 45:1018–1019, 2011. 7, 8

[44] Wil M. P. van der Aalst. Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining. In *AP-BPM*, 2013. 67

[45] Wil M. P. van der Aalst. Extracting event data from databases to unleash process mining. In *BPM*, 2015. 1, 2, 8

[46] Wil M. P. van der Aalst, Arya Adriansyah, Ana Karla A. de Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, R. P. Jagadeesh Chandra Bose, Peter van den Brand, Ronald Brandtjen, Joos C. A. M. Buijs, Andrea Burattin, Josep Carmona, Malú Castellanos, Jan Claes, Jonathan Cook, Nicola Costantini, Francisco Curbera, Ernesto Damiani, Massimiliano de Leoni, Pavlos Delias, Boudewijn F. van Dongen, Marlon Dumas, Schahram Dustdar, Dirk Fahland, Diogo R. Ferreira, Walid Gaaloul, Frank van Geffen, Sukriti Goel, and year=2011 Christian W. Günther and Antonella Guzzo and Paul Harmon and Arthur H. M. ter Hofstede and John Hoogland and Jon Espen Ingvaldsen and Koki Kato and Rudolf Kuhn and Akhil Kumar and Marcello La Rosa and Fabrizio Maria Maggi and Donato Malerba and R. S. Mans and Alberto Manuel and Martin McCreesh and Paola Mello and Jan Mendling and Marco Montali and Hamid R. Motahari Nezhad and Michael zur Muehlen and Jorge Munoz-Gama and Luigi Pontieri and Joel Ribeiro and Anne Rozinat and Hugo Seguel Pérez and Ricardo Seguel and Marcos Sepúlveda and Jim Sinur and Pnina Soffer and Minseok Song and Alessandro Sperduti and Giovanni Stilo and Casper Stoel and Keith D. Swenson and Maurizio Talamo and Wei Tan and Chris Turner and Jan Vanthienen and George Varvaressos and H. M. W. Verbeek and Marc Verdonk and Roberto Vigo and Jianmin Wang and Barbara Weber and Matthias Weidlich and A. J. M. M. Weijters and Lijie Wen and Michael Westergaard and Moe Thandar Wynn, booktitle=Business Process Management Workshops. Process mining manifesto. 1, 8, 10

[47] Wil M. P. van der Aalst, Arya Adriansyah, and Boudewijn F. van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 2:182–192, 2012. 7

[48] Wil M. P. van der Aalst, Vladimir Rubin, H. M. W. Verbeek, Boudewijn F. van Dongen, Ekkart Kindler, and year=2010 volume=9 pages=87-111 Christian W. Günther, journal=Software and System Modeling. Process mining: a two-step approach to balance between underfitting and overfitting. 7

[49] Wil M. P. van der Aalst and Minseok Song. Mining social networks: Uncovering interaction patterns in business processes. In *Business Process Management*, 2004. 7

[50] Wil M. P. van der Aalst, A. J. M. M. Weijters, and Laura Maruster. Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16:1128–1142, 2004. 7

[51] Thomas Vogelgesang and year=2015 Hans-Jürgen Appelrath, booktitle=BPM. Multidimensional process mining with pmcube explorer. 67

[52] Jochen De Weerdt, Manu De Backer, Jan Vanthienen, and Bart Baesens. A robust f-measure for evaluating discovered process models. *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 148–155, 2011. 7