MASTER’S THESIS
Towards Cost-Awareness in Process Mining

B.Sc. Wiebe Ernst Nauta

Principal Supervisor: Dr. Moe Wynn (BPM Group, QUT)
Associate Supervisor: Dr. Boudewijn van Dongen (AIS Group, TU/e)
Examination Committee: Dr. Remco Dijkman (IS Group (IE&IS), TU/e)

Eindhoven, July 2011
Abstract

In business process management, the first step toward improvement of processes is As-Is analysis of existing processes before working towards a To-Be situation. Process mining is part of this As-Is analysis and gives insight into various aspects of processes through the mining of events recorded by a system in the form of an event log. This enables discovery of the process as it actually is. Depending on how much detail an event log contains, observations can be made about the process itself, participating resources, the cases in the process and time aspects. This information is used to improve the process. For many businesses, cost reduction is also an important motivator for change. Since most, if not all, sort of change affects processes within a business, cost reduction and process improvement are inseparable from each other. This work addresses the neglected role of cost reduction in process mining. Or, to put it differently, to support management accounting decisions on cost reduction through the use of cost-aware process mining. Taking into account existing cost reduction techniques within management accounting and extending the process mining tool ProM, this work proposes an architecture to support cost-awareness in process mining, and a first implementation of this architecture.

**Keywords**: business processes, process mining, management accounting, cost reduction
Preface

This master thesis documents the results of my graduation project which is the last phase of the master's degree in Business Information Systems at Eindhoven University of Technology. The project was done within the Architecture of Information Systems research group at this university. The majority of the work took place at the Business Process Management research group at Queensland University of Technology.

Towards the end of my coursework I was more or less aware there was a research group in Australia working closely together with research groups of the W&I and IE&IS departments in Eindhoven. When, in the Autumn of 2010, I mentioned the desire to do the last phase of my studies “abroad”, I could not have dreamed of going there. Going abroad is one decision I am glad I did not make solely on cost.

Foremost, I would like to thank Dr. Moe Wynn for her guidance, patience and both instructive and inspiring supervision which almost swayed me to further my career in academia. I express my gratitude to Dr. Boudewijn van Dongen for initially providing me with many great options for graduation projects and for his guidance. I would like to thank Dr. Remco Dijkman for agreeing to be part of the examination committee and assessing my work.

My experience was made infinitely more fun and rewarding thanks to the people at the diverse and always changing BPM research group, from Arthur to Zhaoxia and everyone else: Chun, Khalid, Ross, Sebastian, Raffaele, Henry, Bo, Chathura, Thomas, Niz, Stefan, Alfredo, Marcello, Samia, Jan, Nasim, Jessica, Stephan, Michael, Ayon, Erwin, Ayed, Glenn, Keith and anyone I forgot to mention.

I would like to thank Wil van der Aalst, Eric Verbeek, and Joos Buijs for providing useful insight and answering my questions in person, through mail, and on the ProM forum.

Last but far from least I thank my family for supporting me when I needed it, close friends back home for forgiving me for letting communication sometimes dwindle down to infrequent signs of life, and my girlfriend Hannah for coming into my life and making all things seem more interesting.

Wiebe Nauta
July 2011
's-Hertogenbosch
# Solution Implementation

4.1 Cost Model Implementation ........................................ 28
4.2 Log Annotation Implementation .................................... 28
   4.2.1 Variable Cost Depending on Data .......................... 30
   4.2.2 Variable Cost Depending on Duration ..................... 30
   4.2.3 Log Event Annotation Restrictions ......................... 31
4.3 Reporting Implementation ......................................... 32

# Conclusions

5.1 Deliverables ....................................................... 35
5.2 Limitations and Future Work ...................................... 36

# Bibliography

A Cost Model
   A.1 Cost Input ..................................................... 40
   A.2 Cost Model .................................................... 43

B XES Cost Extension .................................................. 45

C Case Example
   C.1 Mapping ....................................................... 46
   C.2 Cost Driver with Variable Cost from Data .................. 46
   C.3 Cost Driver with Variable Cost from Duration ............. 47
   C.4 Log Event Annotation ......................................... 47

D Cost Reports
   D.1 Cost per Case ................................................ 49
   D.2 Cost per Cost Type ........................................... 50
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>BPM lifecycle</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>General architecture</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>Activity-Based Costing principle</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Process mining within BPM</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>ProM 6 main interface</td>
<td>11</td>
</tr>
<tr>
<td>2.4</td>
<td>XES standard lifecycle model</td>
<td>13</td>
</tr>
<tr>
<td>2.5</td>
<td>Petri net model of the phone repair process</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>Detailed architecture</td>
<td>18</td>
</tr>
<tr>
<td>3.2</td>
<td>Cost model</td>
<td>19</td>
</tr>
<tr>
<td>3.3</td>
<td>Cost driver</td>
<td>20</td>
</tr>
<tr>
<td>3.4</td>
<td>Cost function</td>
<td>22</td>
</tr>
<tr>
<td>3.5</td>
<td>Cost mapping</td>
<td>23</td>
</tr>
<tr>
<td>3.6</td>
<td>Log annotation</td>
<td>24</td>
</tr>
<tr>
<td>4.1</td>
<td>XES standard lifecycle model with durations</td>
<td>31</td>
</tr>
<tr>
<td>4.2</td>
<td>Cost per case and cost type</td>
<td>32</td>
</tr>
<tr>
<td>D.1</td>
<td>Cost per case</td>
<td>49</td>
</tr>
<tr>
<td>D.2</td>
<td>Cost per cost type</td>
<td>50</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Resources involved in the phone repair process . . . . . . . . . . . . . 14
2.2 Data variables in the phone repair process . . . . . . . . . . . . . . 15
2.3 Costs in the phone repair process . . . . . . . . . . . . . . . . . . . 16
4.1 Costs per cost type calculation for case two . . . . . . . . . . . . . . 33
Listings

3.1 Event log cost annotation for a single cost type ........................................... 26
4.1 Event log cost annotation algorithm ................................................................. 29
A.1 Cost input ........................................................................................................... 40
A.2 Cost model ......................................................................................................... 43
B.1 XES cost extension ............................................................................................. 45
C.1 Mapping ............................................................................................................... 46
C.2 Cost driver with data .......................................................................................... 46
C.3 Cost driver with duration .................................................................................... 47
C.4 Event log cost annotation for event .................................................................. 47
Chapter 1

Introduction

This master thesis is the result of a graduation project done within the emerging research area of Cost-Aware Business Process Management (Cost-Aware BPM). This research area explores how management accounting can be supported and improved through the application of Business Process Management (BPM) principles and tools.

BPM defines a methodology and tools to design, configure, execute and diagnose structured and less structured processes within organizations [4, 25]. Through the iterative application of BPM, processes are improved in terms of quality, flexibility, time and/or cost [22].

Management accounting is a field of management which uses historical financial data to make informed business decisions [21]. Through costing techniques used within management accounting, the cost of any aspect of a business such as a resource, department, process, product or service, can be accurately determined. The cost of each aspect of the business can then be analyzed and compared to the budget providing valuable input for decision making. Examples of such decisions are canceling the production of a product with a higher than expected unit cost, raising the price of a product, or taking no action since a specific costly product increases customer loyalty [6].

Although the two fields of management look at different aspects of an organization, there is a significant opportunity in bridging the two, since anything done through the application of BPM has an impact on processes and resources within an organization which in turn influence cost and therefore management accounting decisions. Through Cost-Aware BPM research, an attempt is made to bridge this gap from the side of BPM by incorporating aspects of management accounting into BPM and making BPM systems “Cost-Aware”.

This project explores how management accounting, in particular the support for application of costing techniques, can be incorporated in the diagnosis phase of the BPM lifecycle. A framework is proposed which uses information from management accounting in combination with information produced during the execution phase of the BPM lifecycle to provide accurate, relevant and timely information for decision making.

Enterprise cost reduction and business process improvement both emerge as important topics in Gartner’s 2011 survey amongst more than 2000 CIO’s [12], ranking at places 3 and 5 respectively.
The project was done within the Architecture of Information Systems group at Eindhoven University of Technology. The majority of the work took place within the Business Process Management group at Queensland University of Technology.

In this chapter the outset of the project is further explained. Section 1.1 describes the context of the project followed by Section 1.2 which describes the main project goals. Sections 1.3 and 1.4 describe the scope and methodology, respectively, and in Section 1.5 the content of further chapters is outlined.

1.1 Thesis Context

The research area of Business Process Management (BPM) is broad and covers any kind of actions taken towards the management and improvement of business processes [25]. Business process improvement happens in an iterative way following a number of phases, shown in figure 1.1 (adapted from [4]).

![Figure 1.1: BPM lifecycle](image.png)

Each of the different phases requires a different kind of professional and different tools. This work focuses on the design, but mostly the diagnosis phase (also called the evaluation phase) which closes the loop. Figure 1.1 shows the diagnosis phase can either be started for the first time following the discovery phase or from a previous iteration following the control phase. In both cases data collection takes place to perform the diagnosis. Collecting and using data to get insight into what is happening within processes is known as As-Is analysis. This kind of analysis is done by professional consultants familiar with business processes and workflow modeling who apply what is called Business Process Redesign (or Re-engineering) [23]. In the last decade the role of information systems became increasingly important within BPM. Workflow Management Systems (WfMS) emerged, and later Business Process Management Systems (BPMS) which provide more comprehensive support for the different phases of BPM. These kind of systems and any other systems to do with processes are categorically known as Process-Aware Information Systems (PAIS) [1, 10]. Ideally these systems support the execution of a process but also planning, design, deployment and any other kind of activity to make an information system “process aware”.

The part of As-Is analysis that collects process information from systems rather than through elicitation from people is known as “process mining”. Process mining and management accounting are two disciplines which this work seeks to bring together.
1.2 Problem Description and Goal

The overall problem addressed in this work is the loss of opportunity from seeing BPM and management accounting as strictly separate fields of management. Because the main beneficiary of bridging these management disciplines is management accounting, the omission in current thinking is addressed from the perspective of management accounting. In [21], p8, the International Federation of Accountants (IFAC) describes the activity of gathering information for the purpose of decision making as follows:

Professional Accountants in Business (PAIBs) who design, use, or collect cost information will typically work with many other parts of an organization to analyze and interpret this information for decision making. In most cases, they will need to delve below the level of detail recorded in the financial ledgers and required for external financial reporting.

The mentioned level of detail in the financial ledgers is aggregated cost information. To delve below this level of detail and beyond bookkeeping, an accountant needs information from different parts of an organization. If the organization uses a PAIS which records historical information about what is happening during the execution of the organization’s processes, the information the accountant needs is available in some form. However, the management accountant cannot make decisions based on the information in this form and has to cooperate with professionals from other disciplines to convert any information obtained from the information system to a form useful for management accounting. This problem is summarized as follows.

**Problem summary:** Information gathering for management accounting requires significant effort from the management accountant and other professionals supplying the information. The historical information produced by PAISs usually contains no cost information. If it does contain cost information it is tightly coupled to the system in the form of implemented business rules.

Provided that information produced by a PAIS represents reality and contains sufficient detail, it can be used to derive cost information in its most detailed form, thereby linking what has happened in reality directly to cost associated with what has happened. What is still missing are cost amounts associated with what has happened. This information to do with budgeting is usually present in organizations in ERP, HRM, CRM and/or other kinds of information systems. Examples of this are the cost of hourly wages, cost of hourly machine usage, per unit cost of consumable materials, or fixed costs associated with specific activities which are done.

By combining the dynamic information of things that have happened and are captured by a PAIS with the more static information about the cost of each of these things, the exact cost of historic information can be created in detail. This fine-grained cost information can then be aggregated in various ways to provide support for making decisions related to management accounting. This vision is represented in as a general architecture overview shown in Figure 1.2. The central layer labeled “Process Mining” takes in information from a PAIS (in this case a BPMS) and information from management accounting to produce information for making decisions. The form and content of the various informational input and output is further explained in Chapter 3.
Within the part labeled “BPMS” in Figure 1.2, three separate steps are distinguished. These steps correspond to different moments in time when new information becomes available to be used by an implementation of the architecture. In the design phase of the BPM lifecycle, the process model, organizational model, and cost amounts associated with possible things that can happen are available. The process model is assumed to be the actual BPMS executable process model rather than an ideal or as-it-should-be process model. These static informational inputs are combined into what throughout this work is referred to as a “cost model”. After a process has executed for some time, information is available in the form of an “event log”. This log is enhanced or annotated by computing the cost of what has happened using the cost model. Finally, the detailed cost information in the annotated log is queried in various ways producing reports for the benefit of making decisions. The cost model contains functions for defining the contents of the reports. The layout is defined by the management accountant in the form of a report template. The general architecture is the basis of the solution for the problem defined previously, and is reflected in the goal of this work which is summarized below.

**Goal summary:** Design an architecture to make process mining cost-aware, thereby supporting the information gathering phase and the decision making phase in management accounting.

Based on this goal and the described architecture, a number of sub goals are defined. The sub goals relate to the components proposed in the general architecture to solve the defined problem.

**Sub goal 1:** Define in a flexible format aspects which drive cost and aspects required for cost reporting and link these to concepts found in BPM.
Sub goal 2: Find a structured way to incorporate cost information in event logs from a PAIS.

Sub goal 3: Create an implementation using the flexible format to enhance event logs with cost information.

Sub goal 4: Generate reports from the event logs to support decision making in management accounting.

1.3 Research Scope

The main focus of this work is bridging management accounting and the diagnosis phase in BPM in order to support decision making. The initial scope is set by the research goal and sub goals. Through a detailed architecture design and an implementation of this design, decision making support is realized. The implementation is illustrated through a fictional test case example.

It should be noted process mining is a broad field and this work is restricted to cost-awareness in process mining. Its support of management accounting is in the form of detailed cost reports. However, many more aspects of process mining for supporting management accounting can be conceived of. Two are noted specifically.

First, visualization of cost associated with process execution in the form of charts and graphs showing cost from different perspectives gives a fast overview of cost incurred during process execution. Visualization is not specific for cost reporting and is an important part of process mining in general [9, 20]. Secondly, prediction of cost during process execution based on historic data of previous executions is desirable. Prediction enables anyone involved in the process to make decisions while the process is being executed rather than in the diagnosis phase. Since cost is an important factor in decision making, real-time information about cost based on historic data could save future cost. Like visualization, prediction is a part of process mining in general [5, 8].

1.4 Research Methodology

To satisfy the research goal the project underlying this work is divided into several phases. Although there is an amount of overlap between the phases, the phases are listed in order of start time.

Literature study A literature study is conducted to gain insight into the field of management accounting and the current developments and practice in this field. Three costing techniques are analyzed in terms of informational input and informational output of each technique. All are supported through this work. Knowledge of process mining is a prerequisite for the work.

Modeling and architecture design From the literature study, requirements for decision making support are captured in conceptual models. Taking into account the implementation environment, a detailed architecture is proposed together with data input and output formats.
**Incremental implementation phase** A pilot of the core log annotation and reporting algorithms is implemented in the implementation environment. Together with the conceptual and data models the algorithms are incrementally refined to cover the core functionality.

**Proof of concept through simulation data** Instances of fictitious cost models covering the most common cost elements used in process execution are created. These cost model instances together with event logs obtained through process simulation of a test case example are then used as input for the implemented algorithms.

### 1.5 Outline

In Chapter 2 preliminary concepts in management accounting and process mining are discussed, as well as the conceptual modeling language ORM 2 and the test case example. Chapter 3 discusses the design of the proposed solution, followed by Chapter 4 where the implementation of the design is described. Chapter 5 highlights the main conclusions of the work and describes its limitations and future work.
Chapter 2

Preliminaries

2.1 Management Accounting

One of the ways organizations can be more profitable is by reducing enterprise cost. The difficulty comes from the question of what to reduce without affecting the profitability of the business more than what is gained by the cost reduction.

In an organization where all cost can be attributed directly to a product, it is easy to make decisions about cost reduction. Products which cost more to produce than their revenue worth are candidates for discontinuation, assuming their only value to the business is sales revenue. Up to three decades ago, in traditional manufacturing companies, decisions about cost reduction could be made this way. Nowadays, most often the indirect cost of business functions like administration, marketing, sales, distribution, IT are greater than direct production cost. (Business functions unrelated to product cost such as R&D are ignored.) This gives rise to a desire to measure these overhead costs as well, and attribute them directly to a product or service, assuming all of the business activities performed in these business functions are ultimately to produce a product or deliver a service [6, 7]. Various costing techniques based on the notion of activities have been developed. In this section an overview of three techniques is given, concluding with the extent of support of these techniques delivered through the approach in this work. The techniques are Activity-Based Costing (ABC), Time-Driven ABC (TDABC), and Resource Consumption Accounting (RCA) which are all based on the notion of activities. Fictional and practical examples of applying the costing techniques in practice are given in referred work, a comprehensive overview of the field is left for further reading\(^1\).

2.1.1 Activity-Based Costing (ABC)

Since activities are the link between the indirect cost of overhead business functions and the production of products or delivery of services, they are central to the earliest

\(^1\)http://web.ifac.org/publications/professional-accountants-in-business-committee

\(^2\)http://www.imanet.org/resources_and_publications/management_accounting_quarterly.aspx
efforts to solve problems of traditional costing systems. The first conceived costing technique is Activity-Based Costing (ABC) [6, 7]. This costing technique first identifies all activities contributing to the production of a product or delivery of a service. Next, it considers the per unit cost of each resource in each business function (both product- and overhead business functions) and to what extent the resource is involved in the realization of a product or service. After that, cost can be aggregated by activity. The next step is identifying how much each activity contributes to the realization of each product or service. Finally, the per unit consumption of a resource per unit of each product or service is discovered, thus obtaining the cost per product or service. These steps are shown in Figure 2.1.

Figure 2.1: Activity-Based Costing principle

After being put in practice a number of flaws in the ABC costing technique were discovered. The first is that information gathering and keeping this information up-to-date in itself is costly [17]. The information needs to be updated with every change in activities and resources. Moreover, measuring resource involvement in activities through surveys can be inaccurate. Secondly, complex, granular activities defined as a single activity in an ABC model are abstractions of reality making the costing technique less accurate. Splitting these activities up leads to more effort to capture them in an ABC model [11]. Lastly, excess capacity of resources is not accounted for [24]. However, at the same time referred work acknowledges the principles of ABC are still widely used as an alternative to traditional bookkeeping or an extension thereof.

2.1.2 Time-Driven ABC (TDABC)

Whereas the ABC costing technique looks at cost from the perspective of resources and spreads this cost first over activities and then over products and services, Time-Driven ABC (TDABC) takes a different approach solving a number of the aforementioned problems of ABC. TDABC defines drivers as “resource-activity drivers” which capture in a detailed way how resources are involved in activities for the purpose of delivering a specific product or service. All measurements are in time units.
hence the time-driven aspect of this costing technique.

Information gathered for TDABC is more granular, therefore gathering and updating this information seems to be more effort compared to ABC. However, it is less dependent on employee surveys and can be updated with less effort using information from ERP, HRM and CRM systems [17]. It takes into account variations of the same activity depending on which product or service the activity is performed for leading to more accurate cost accounting [11]. Since cost is attributed to performance of activities in a detailed way, this actual cost can be compared to budgeted cost for resources yielding the cost of idle resources [24].

2.1.3 Resource Consumption Accounting (RCA)

Like ABC and TDABC, Resource Consumption Accounting (RCA)\(^3\) uses activity-based cost drivers. Unlike TDABC it is proposed as a comprehensive accounting technique and allows for attributing resource costs directly to products, fixed and proportional cost for resources and resource pools, and the notion of cost types [24]. The technique stresses the importance of the principle of causality with “fundamental operations transactions as the primary source for financial and quantitative data [...] without the cost distortions and complexity of inappropriate allocations of cost”\(^4\).

RCA is a relatively new technique. The International Federation of Accountants (IFAC) has placed it as one of the more mature techniques in its maturity model [21].

The commonality of all mentioned costing techniques is a form of activity driver for allocating direct and indirect cost to operations producing products or delivering services. While the ABC and TDABC costing techniques primarily focus on allocating cost of resources to products and services, the RCA costing technique provides more ways of allocating cost through direct allocation and use of responsible managers and cost types. This work strives to provide a generic method and tools to support all mentioned costing techniques.

2.2 Process Mining

Process mining is a maturing research area and is part of the diagnosis phase within BPM. During the execution of a process, a Process-Aware Information System (PAIS) logs information about what is happening in the form of an event log [1, 10]. The commonality of these event logs is that each event log contains a chronological list of events. Depending on the system, the event logs can be in various formats containing particular information about the occurred events. Depending on how much information is logged in each event, the event logs can be analyzed from four different perspectives (or dimensions) listed below.

\(^{3}\)http://www.rcainstitute.org/rcai-3-WhatIsRCA.php
\(^{4}\)http://en.wikipedia.org/wiki/Resource_Consumption_Accounting
Control flow The process being executed in the PAIS.

Organizational The resources participating in the process (human or non-human).

Case What is executed in the current process (also process instance).

Time The chronological order.

Figure 2.2 (adapted from [3]) shows how process mining uses information recorded through a PAIS in the form of event logs. Using event logs, a large number of analysis techniques and tools can be applied which fall into three categories: discovery, conformance and enactment. Although these categories describe techniques and tools for discovering process models, checking conformance of process models and using process models for enactment support, the three categories can be applied to cost as well. This work focuses on discovery of cost incurred during process execution.

2.2.1 ProM

Process mining is enabled through software. The implementation environment for this work is process mining tool ProM\textsuperscript{5,6}. Currently two very different versions of the software are in use, each having their own advantages and disadvantages. This work is implemented in ProM version 6 and although it is an overhaul of its predecessor and still in an incubation phase, benefits of ProM 6 will be highlighted.

The tool is set up in a modularized way composed of a core framework and “plug-ins”. The core framework provides the main user interface, shown in Figure

\footnotesize{\textsuperscript{5}http://www.promtools.org/prom5/}
\footnotesize{\textsuperscript{6}http://www.promtools.org/prom6/}
and defines how plug-ins can be added, invoked, and maintained within the framework. In the main user interface an “object pool” of data objects forms the basis for invoking plug-ins. Plug-ins can be of four types listed below.

- **Import** Import data objects into the object pool.
- **Export** Export data objects from the object pool.
- **Default** Convert one set of objects into another set of objects.
- **Visual** Visualize an object.

Default plug-ins are not necessarily self-contained and may depend on software libraries or other plug-ins to complete their task. Apart from invoking plug-ins on data objects from the user interface, ProM 6 plug-ins can be invoked on an instance of ProM 6 over a network. This way, process mining is performed in a distributed fashion and enables grid computing. Automating and performing of process mining operations in batches is realized through chaining (or stacking) of plug-ins making the often needed sequential execution of a series of plug-ins less cumbersome to perform.

An addition to the ProM 6 framework that is envisioned but yet to be implemented is operational support for decisions to be made during the execution of a process through real-time process mining. Through a “dashboard” showing what is happening in cases executed in the process and comparing these case to historic information, predictions can be made about the best options in terms of time and cost for decisions within the cases [2]. Since cost is often a consideration when making decisions, this feature makes ProM 6 a promising implementation environment for
this work.

2.3 Event Logs

Event logs are the required input for process mining. Next, the event log format used with the ProM 6 implementation environment is described, followed by how the format accommodates the different stages in tasks having a duration through a task lifecycle model.

2.3.1 Extensible Event Stream (XES)

This work uses the recently developed Extensible Event Stream (XES) event log format [13]. It is a generic, XML-based format for capturing event information in a structured way and used mostly in the context of process mining. Contrary to its predecessor, Mining XML (MXML), no assumptions are made about the information stored in events. The XES format uses the dimension of a case (or trace in XES) for grouping events. The time dimension, control flow dimension and organizational dimension are implemented as optional extensions of events in a case. In this work, extensibility of XES event logs is used for annotating events with cost information conforming to a cost extension of the XES format. Extensions to the event log format are defined in the XES extension format (XESEXT). This format defines the format of XES extensions.

In unison with XES a number of libraries have been implemented in ProM for convenient access to XES event log information [14]. The libraries also provide backward compatibility for the MXML format. A number of academic and commercial tools are available to convert various data sources to the XES event log format\(^7,8,9\).

2.3.2 Standard Lifecycle Model

Tasks in process can be atomic or have a duration. Whenever a task is performed for a case and needs a resource to be processed it becomes a “work item” or “task instance”. Task instances of tasks which have a duration can go through a number of states while being processed, each state determining the status of the task instance. If a PAIS logs this information, the state changes will be logged as events in the event log.

Apart from the four mentioned dimensions of processes and event logs, the XES event log format also provides a lifecycle\(^10\) extension which formalizes the allowed order of state changes of a task instance. The lifecycle extension is shown in Figure 2.4 (taken from [13]). It should be noted that in practice task instances follow a subset of this lifecycle, e.g. a start event followed by a complete event, or conform

---

\(^7\)http://www.promtools.org/promimport/
\(^8\)http://www.processmining.org/xesame/start
\(^9\)http://fluxicon.com/nitro/
\(^10\)The term “lifecycle” throughout this work refers to the XES standard lifecycle for work items, unless it is used as “BPM lifecycle”. 

12
to a less expressive or more expressive lifecycle. In this work the lifecycle is used for calculating variable cost depending on durations.

Figure 2.4: XES standard lifecycle model

2.4 Test Case Example

To illustrate the way the goal defined in Section 1.2 is reached, a test case example is used throughout Chapters 3 and 4. The case example itself is described here, all examples throughout this work refer to this test case example.

For this work, no real world example of a process together with event logs and management accounting data are available. Therefore, a simulated test case example is used. The test case example is a simulation of a phone repair process\textsuperscript{11}, and management accounting data is defined for this process. Event logs produced by way of executing a simulation of this process contain no noise or unexpected behavior. This case example is selected because the process simulation produces event logs having all characteristics needed to validate the work comprehensively. The necessary characteristics are listed below.

- Parallel, choice and loop constructs
- Tasks with a duration defined by a work item lifecycle

\textsuperscript{11}http://www.processmining.org/prom/tutorials
• Resources participating in process execution
• Case data produced by random variables
• Control flow decisions based on case data values

Figure 2.5: Petri net model of the phone repair process

The phone repair process is shown in Figure 2.5. As is clear from the model, the process contains the named constructs and four tasks having a duration. Furthermore, there are twelve different resources in three roles participating in the process which are listed in Table 2.1 and four case data variables listed in Table 2.2. The four case data variables contain information about the phone type, “phoneType”, how severe the phone damage is, “defectType”, the number of performed repairs, “numberRepairs”, and whether the last repair was successful, “defectFixed”.

<table>
<thead>
<tr>
<th>Role</th>
<th>Quantity</th>
<th>Naming Scheme</th>
<th>Task(s) Assigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tester</td>
<td>6</td>
<td>Tester1, ...</td>
<td>Analyze Defect, Test Repair</td>
</tr>
<tr>
<td>SolverS1</td>
<td>3</td>
<td>SolverS1, ...</td>
<td>Repair (Simple)</td>
</tr>
<tr>
<td>SolverC1</td>
<td>3</td>
<td>SolverC1, ...</td>
<td>Repair (Complex)</td>
</tr>
</tbody>
</table>

Table 2.1: Resources involved in the phone repair process

The costs associated with each task, resource, and data variable are shown in Table 2.3. It is possible to interpreted each row in this table intuitively but the table
Table 2.2: Data variables in the phone repair process

<table>
<thead>
<tr>
<th>Data variable</th>
<th>Domain</th>
<th>Initializing Task(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>phoneType</td>
<td>T1, T2, T3</td>
<td>Analyze Defect</td>
</tr>
<tr>
<td>defectType</td>
<td>1..10</td>
<td>Analyze Defect</td>
</tr>
<tr>
<td>numberRepairs</td>
<td>1..5</td>
<td>Test Repair, Archive Repair</td>
</tr>
<tr>
<td>defectFixed</td>
<td>true, false</td>
<td>Test Repair, Archive Repair</td>
</tr>
</tbody>
</table>

will be further explained in Chapter 3 where each table row is defined as a “cost driver”\(^{12}\) explaining how cost is exactly linked to the test case example process.

The phone repair process itself is described in short. When a broken phone is brought in by a customer it is first registered. Then it is analyzed for defects by a tester determining the phone type and the defect type. Depending on the severity of the defect a simple or a complex repair is performed by solvers. The number of repairs is set to one. After the repair, the phone is tested by a tester determining whether the phone is fixed. If it is found to be fixed, the details of the repair are ready for to be archived and the phone can be returned to the customer. If it is still broken, the repair is restarted and the repair and testing are repeated. A maximum of five repairs can be performed on a single phone. If, after five repairs, the phone is still broken, the repair details are archived and the phone is returned to the customer. In parallel with the repair and the testing, the customer (“User” in the petri net) is informed about the severity of the defect. The customer has to have been informed before the repair can be archived.

The process simulation is executed using Colored Petri Net Tools (CPN Tools)\(^{13}\) producing an event log with 1000 completed cases. CPN Tools supports the use of case data (color), and timed simulation using discrete time steps. The produced event log is composed of multiple text files each containing information about the events of a single case. This event log is converted into a log in MXML format using ProM Import\(^{14}\) and subsequently converted into XES format using ProM 6. Note that the routing in the process is dependent on case data. In the simulation model, transitions have guards which are not shown in the petri net in Figure 2.5 for reasons of simplicity. Furthermore, resources “Tester” 1 through 6 have the same working speed and always succeed. Resources “SolverS” 1 through 3 and “SolverC” 1 through 3 have varying working speeds (Solvers S and C 1 are the fastest) and a varying chance of a successful repair (Solvers S and C 1 have the lowest chance of a successful repair). This is significant for reporting of incurred cost of busy resources.

\(^{12}\)The term “driver” is borrowed from the domain of management accounting and used in costing techniques

\(^{13}\)http://cpntools.org/

\(^{14}\)http://www.promtools.org/promimport/
<table>
<thead>
<tr>
<th>Cost Type</th>
<th>Task</th>
<th>Resource / Role</th>
<th>Data Variable</th>
<th>Data Value</th>
<th>Cost</th>
<th>Unit</th>
<th>Duration Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wages Tester</td>
<td>Analyze Defect</td>
<td>Tester</td>
<td></td>
<td></td>
<td>60,-</td>
<td>Hour</td>
<td>Busy</td>
</tr>
<tr>
<td>Wages Tester</td>
<td>Test Repair</td>
<td>Tester</td>
<td></td>
<td></td>
<td>70,-</td>
<td>Hour</td>
<td>Busy</td>
</tr>
<tr>
<td>Wages Solver</td>
<td>SolverS</td>
<td></td>
<td></td>
<td></td>
<td>65,-</td>
<td>Hour</td>
<td>Busy</td>
</tr>
<tr>
<td>Wages Solver</td>
<td>SolverC</td>
<td></td>
<td></td>
<td></td>
<td>75,-</td>
<td>Hour</td>
<td>Busy</td>
</tr>
<tr>
<td>Facilities</td>
<td>Analyze Defect</td>
<td></td>
<td></td>
<td></td>
<td>10,-</td>
<td>Hour</td>
<td>Working</td>
</tr>
<tr>
<td>Facilities</td>
<td>Test Repair</td>
<td></td>
<td></td>
<td></td>
<td>15,-</td>
<td>Hour</td>
<td>Working</td>
</tr>
<tr>
<td>Parts T3</td>
<td>Analyze Defect</td>
<td>phoneType</td>
<td>T3</td>
<td></td>
<td>20,-</td>
<td>Invocation</td>
<td></td>
</tr>
<tr>
<td>Admin.</td>
<td>Test Repair</td>
<td>defectFixed</td>
<td>false</td>
<td></td>
<td>10,-</td>
<td>Invocation</td>
<td></td>
</tr>
<tr>
<td>Failed Repair</td>
<td>Archive Repair</td>
<td>numberRepairs</td>
<td>5</td>
<td></td>
<td>200,-</td>
<td>Invocation</td>
<td></td>
</tr>
<tr>
<td>Analysis Complexity</td>
<td>Analyze Defect</td>
<td>defectFixed</td>
<td>false</td>
<td></td>
<td>1,- *</td>
<td>defectType</td>
<td>Invocation</td>
</tr>
<tr>
<td>Analysis Complexity</td>
<td>Analyze Defect</td>
<td>defectType</td>
<td>0,01 *</td>
<td>defectType</td>
<td>Minute</td>
<td>Working</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.3: Costs in the phone repair process
Chapter 3

Solution Approach

Cost-Aware BPM and the goal of this work have been described in Chapter 1 and the implementation environment has been described in Chapter 2. The goal is realized within the implementation environment through the design of an architecture described in this chapter. The implementation itself is described in Chapter 4.

As mentioned, a BPMS should ideally support all phases of the BPM lifecycle. This work restricts itself to cost aspects of the design and diagnosis phase. The architecture proposed in this chapter uses information from the BPMS for these two phases. As in BPM, management accounting uses historic information for making future improvements. Another similarity is that both fields view business operations from the perspectives of tasks in the process, resources and case data. The similarities are important factors in this work enabling the fields to be related through models and architecture designs.

3.1 Architecture Overview

The focus of this work is bridging management accounting and process mining conceptually whereby the design of the cost model and annotation of event logs are the core parts of this work. As seen in Figure 1.2, the general architecture consists of three components. Not all components of the general architecture have been fully implemented and parts have been left as future work. In this section the general architecture will be further specified in the context of the implementation environment detailing the completed and yet to be completed parts.

In Section 2.2.1 process mining tool ProM 6 is discussed as a possible implementation environment. Apart from the benefits listed in Section 2.2.1, ProM 6 is in active development and will expand as research into process mining progresses whereby ProM 6 will be the main implementation environment where theories and conceptual work are put to the test. This work contributes to the development of ProM 6 and is very likely to benefit from future development in ProM 6. Because of ProM 6’s modular setup, the three components in the general architecture can each be split up into multiple “plug-ins”. Of the four plug-in types, import, export, default and visual, default plug-ins are generally responsible for the core of the work each taking in a set of input data and transforming this into output data, possibly using other plug-ins. The general architecture components have been split into
ProM 6 plug-ins where each component corresponds to a default plug-in.

In Figure 3.1 the detailed architecture is shown. As is clear from the detailed architecture, creation of the cost model is not implemented as are parts of the reporting plug-in. One necessary additional plug-in “Calculate durations” has been added. This plug-in calculates time aspects of the event log and is indirectly related to cost analysis since variable cost can depend on durations. Descriptions and implementation considerations of the individual plug-ins shown in the detailed architecture are further discussed in Chapter 4.

3.2 Cost Model Design

As seen in the general and detailed architecture diagrams, creation of a cost model requires information from management accounting and from a BPMS. Since the cost model uses static information from management accounting and the BPMS, it is designed independently from process execution and can be created at any point after the design stage in the BPM lifecycle. The cost model contains all necessary information for annotating an event log with cost information. Figure 3.2 shows the conceptual cost model, modeled using the Object-Role Modeling version 2 (ORM 2) notation.

For readers unfamiliar with this modeling notation, a short summary is provided to understand the models. The complete ORM 2 notation is given in [16]. The ORM 2 data modeling language captures semantics by relating entity types

\footnote{http://www.orm.net/}
Entities are similar to the concept of attributes in relational database theory and fact types can be compared to non-normalized relations in relational database theory. Furthermore, ORM 2 uses the concept of inheritance by way of generalization and specialization similar to Unified Modeling Language (UML).

Fact types join one to many entities specifying uniqueness constraints through lines above role boxes which make up the fact type. Each entity having a role box in a fact type may have a mandatory role in the fact type denoted by a solid dot at either the entity end or its role box end in the fact type. Uniqueness constraints implicitly specify the cardinality of each entity in the fact type. Fact types all have a predicate which describes the relation of the entities related through the fact type. Constraints which can not be modeled by way of the structural elements of ORM 2 are described in textual constraints in footnotes below the model. ORM 2 is
regarded as an intuitive and expressive data modeling language [15]. The modeling notation has been compared extensively to other modeling techniques [18].

Although the fact types in the model determine direct relationships between entities, textual constrains are necessary to capture requirements where possible values in the domain of entities are related indirectly. The cost model entity is related in three fact types with predicate ‘contains’ to the three entities listed below.

**Cost Driver** Defines how cost is associated with a process and the cost amount incurred per unit of activity or time unit.

**Cost Function** Defines the content of a cost report.

**Mapping** Relates terms in management accounting to terms used in the BPMS.

The three entities are each described in more detail in the following sections.

### 3.2.1 Cost Drivers

![Diagram of Cost Driver](image)

**Figure 3.3: Cost driver**

Cost drivers are defined by a management accountant and are part of the “Cost input” in the detailed architecture. (Figure 3.3 shows the cost driver from the cost model.) A cost driver captures cost associated with workflow elements of type task, resource or case data incurred during the execution of a process. Furthermore, a cost driver has an id, a name, and possibly a description. The entities a cost driver is associated with fall into the categories of management accounting concepts and workflow concepts. Each entity directly related to the cost drivers is further described below.
**Cost Technique Type** The optional cost technique type entity identifies the costing technique associated with the cost driver. The cost drivers for multiple costing techniques can be specified in a single cost model.

**Workflow Element** This entity determines either the task, or one of the resources or case data variables a cost driver is associated with. A cost driver has to be associated with at least one workflow element of which at most one of type task, multiple of type resource and multiple of type case data. However, all other combinations of associating workflow elements with cost drivers are possible. (Constraint 4 in the cost model.) This flexible, intuitive approach allows for a diversity of scenarios for cost association. Workflow elements associated with a cost driver determine whether a cost driver incurs cost for a certain task invocation. Selection of cost drivers for determining the cost of task invocations is further described in Section 4.2.

**Money** The cost per unit having a monetary value and a currency. Section 4.2.1 describes the implementation of the algorithm based on input from a cost model showing the cost amount can also be an expression using case data values.

**Unit Type** A unit specifying either the cost of a task invocation or the cost per unit of time for a certain duration of the task invocation. The current implementation for event log annotation using the cost model supports minute and hour.

**Duration Type** A task invocation can be composed of multiple durations, each having a type. If the unit type is a unit of time the duration type should be specified, else the duration is ignored in the log annotation implementation using the cost model. (Constraint 2 in the cost model.) The duration type associated with the cost driver is either a task duration type or a resource duration type. (Constraint 5 and 6 in the cost model.) The various duration types and their meaning are further described in Section 4.2.2.

**Cost Type** Cost drivers are associated with one or more cost types. A cost type is a label representing the sort of cost the cost driver incurs and is used in cost reporting. For example, the cost type can determine whether the cost is fixed or variable, associated with department A or B, or any other criterion to group cost in a cost report.

Together, the six entities define how cost is incurred during process execution. Furthermore, a workflow element of type case data has an optional attribute “WE Value” which is used to determine if the cost driver is applicable for attributing cost to a task invocation having the same case data variable with the same value as specified in the workflow element. (Constraint 3 in the cost model.) If the value is not present for the case data variable, the variable name can be used in the cost amount expression of the cost driver. (Which is further explained in 4.2.1)

The concept of a cost driver is illustrated through a number of examples listed below. The examples are based on the test case example in Section 2.4.
• The variable hourly cost of resource “Tester1” working on task “Analyze Defect” is 60,- AUD

• The variable hourly cost of resource “SolverS1” is 65,- AUD

• The variable hourly cost of performing task “Analyze Defect” is 10,- AUD

• The fixed cost for task “Analyze Defect” where a phone of type “T3” is analyzed is 20,- AUD

• The fixed cost for task “Archive Repair” where a phone is repaired five times and is not fixed is 200,- AUD

• The variable cost for task “Analyze Defect” is 1,- AUD to 10,- AUD, depending on the severity of the phone damage “defectType”

• The variable cost for task “Analyze Defect” is 0,01 AUD to 0,1 AUD per minute depending on the severity of the phone damage “defectType”.

Cost drivers specify how cost is incurred during process execution at the most elementary level of a task invocation. Next, cost functions are described which define how cost is aggregated for the purpose of cost reporting.

3.2.2 Cost Functions

Cost functions define the content of a cost report and, like cost drivers, are part of the “Cost input” in the detailed architecture. (Figure 3.4 shows the cost function from the cost model.) A cost function, together with a cost annotated event log and a report template defining the lay-out of the cost report form the input for

2All currencies are in Australian Dollars (AUD)
cost reporting. From the cost annotated event log the cost function selects and aggregates information resulting in a data set which is displayed by applying the lay-out in the report template. Information from the cost annotated event log is selected and aggregated though parameters defined in the cost function. The actual function is defined in an expression associated with the cost function which uses the parameters to produce the data set for the cost report. Three examples of cost function output are listed below.

- List of Cost per Case
- List of Cost per Cost Type
- Table of Cost per Case and Cost Type

The cost reports present the cost of the process from various perspectives and are useful for decision making. Cost reporting is further discussed in Section 4.3.

3.2.3 Mappings

Upon introducing the concept of a cost model, it was mentioned that the cost model describes information used in management accounting and in a BPMS. However, the fields of management accounting and BPM need not necessarily use the same terminology. In most cases in practice, the BPMS is defined and configured by business professionals and engineers who use one terminology, and management accountants use a different terminology linked to accounting, budgeting, and external reporting. To resolve this, a requirement could be added asserting that both the input from the BPMS and the management accounting input are defined using the same terminology. Or the cost model could contain a mapping which maps the different terminologies. This last solution lets both fields keep their terminology and was chosen in this work.

![Figure 3.5: Cost mapping](image)

\[1\] Each mapping should have an identifier of the type cost and an identifier of the type workflow
Through the use of mappings, different terms used in the BPMS and in management accounting which describe the same task, resource, or case data variable are related. (Figure 3.5 shows the mapping from the cost model.) For the purpose of creating the mappings in the cost model, two types of information are necessary, namely the process model and the organizational model used in the workflow engine within the BPMS, and management accounting information detailing cost associated with the tasks, resources and case data involved in the execution of the process which the BPMS supports. Mappings are required for every workflow element (task, resource or case data variable) related to a cost driver.

With the definition of cost drivers in a cost model and terminologies of the two fields of management fully mapped, the cost model is used to enrich historical information of process execution with cost information.

### 3.3 Log Annotation Design

After a process has been executed for a number of cases, an event log is available. Event logs and extension of event logs have been described. Now, a way is proposed to extend an event log with cost information. A cost model and an event log form the input for annotation an event log with cost information.

The ORM 2 representation for annotation an event log with cost information is shown in Figure 3.6. It combines entities from BPM with entities from the cost model. The entities “Case”, “Task” and “Resource” correspond to the different kinds
of workflow elements in the cost model in Figure 3.2. “Cost Type” and “Money” are present in the cost model, as well as the two duration types “Task Duration Type” and “Resource Duration Type”. All of these entities in the cost model and the log annotation model denote the same information. Whenever a task has been performed for a case, a “Task Instance” is available in the event log. Cases can have associated case data which is not shown in Figure 3.6. A task instance can have zero or more resources associated with the task instance at different points in time.

Both task instances and the associations between resources and task instances can in turn be associated with a duration. Since an event log contains all information about time aspects of task instances and time aspects of associations between resources and task instances, durations can be calculated for both and used in cost calculations. Because task instances can be interrupted, the actual duration may be different from the start time subtracted from the end time and the entity “Duration” has a value duration for calculating the duration itself. Durations are derived from the event log but not stored. As specified in the detailed architecture in Figure 3.1, calculation of durations is implemented as a separate plug-in, therefore it is possible to execute duration calculation separately and to view and store the results. Examples of task durations are shown below.

- Working time for completed task “Analyze Defect” for case 1 lasted 10 minutes from 8:00:00 to 8:10:00
- Waiting time for canceled task “Test Repair” for case 2 lasted 8 minutes from 12:34:00 to 12:42:00

Examples of resource durations are as follows.

- Tester3 was busy with completed task “Analyze Defect” for case 1 for 7 minutes from 12:23:00 to 12:30:00
- Tester6 was assigned to canceled task “Test Repair” for case 10 for 6 minutes from 11:42:00 to 11:48:00

The cost annotation itself is an association between a task instance, cost type, and a monetary value together with its currency. The cost annotation is derived from the cost model and event log and stored in the event log. Three examples of cost annotations of a task instance “Analyze Defect” performed by resource “Tester4” on a phone of type “T3” are shown below.

- Variable wages of “Tester4” performing task “Analyze Defect” is 10,- AUD (60,- AUD per hour for 10 minutes)
- Variable facility cost for task “Analyze Defect” is 1,67 AUD (10,- AUD per hour for 10 minutes)
- Fixed cost for analyzing a phone of given type is 20,- AUD (fixed per task instance and phone type)

The cost of this task instance comes from three different cost types of which two are variable and one is fixed. Each of the cost type annotations for the task instance
come from a separate driver. The key values in the cost annotation identify the cost types and the drivers. Listing 3.1 shows the cost annotation for the cost type “Parts T3” for the final event of the task instance.

```
<float key="cost:type:Parts T3" value="20.0">
  <float key="0df9efb7-5ab8-4695-8d66-3d4a46d387d9" value="20.0"/>
</float>
```

Listing 3.1: Event log cost annotation for a single cost type

The complete log event cost annotation of this example is included in Appendix C.4. The format for log event annotation is given in Appendix B. This format is based on the extensibility of the XES event logs described in Section 2.3. The process itself, cases and events can all be associated with a currency. Both cases and events can have cost associated per cost type. Nesting of information is possible in the XES events logs. Per cost type, multiple cost driver id’s incurred costs for each cost driver are nested detailing which cost drivers account for the cost per cost type.

Cost annotations having this level of detail form the basis for cost reporting. Section 4.2 explains how annotation of the event log with cost information is implemented.
Chapter 4

Solution Implementation

In Chapter 3, designs, input formats and output formats were discussed as well as examples of each taken from the test case example. In this chapter the implementation of the conversion of data defined in the respective formats will be described.

The general architecture has further been specified as a detailed architecture, shown in Figure 3.1, for implementation in the implementation environment ProM 6. The top half of the detailed architecture shows the ProM 6 cost architecture plugins, the bottom half shows the input data from the BPMS, management accounting and created data under “Cost objects”. All types of data have a particular format. The first plug-in “Create cost model in UI” is designed to create a cost model from both BPMS and management accounting input data guided by a user through a user interface. Here, a “user” can be a BPM specialist or management accountant. The second plug-in, “Create cost annotated logs”, annotates an event log produced by the BPMS with cost information using the created cost model. This cost-annotated event log can be analyzed through process mining like any other event log. A “Cost Reporting” plug-in has been designed to produce reports from the cost-annotated event log which is discussed in Section 4.3.

In this work, no implementation for plug-in “Create cost model in UI” has been created. This plug-in takes in the process model and organizational model from the BPMS and cost drivers and functions defined by a management accountant for the purpose of reviewing the cost input data, refining the cost functions, and creating a mapping between terms. This plug-in could optionally be expanded to take in management accounting information and converting this to cost drivers and cost functions.

The XML schema definitions for the cost input and the cost model can be found in Appendix A and the XES cost extension format is included in Appendix B. The implementation described in this chapter has been validated using the phone repair process example described in Section 2.4. Parts from the XML input files associated with this example are included in Appendix C. (The complete input file is too large to include as a whole.) Cost reports from the example are included in Appendix D.
4.1 Cost Model Implementation

The cost model is defined conceptually and in the form of a data model. To define cost drivers in a cost model, information is needed from various systems in an organization. The cost model has been split up in parts provided by management accounting and information provided through a BPM professional. Cost information provided by the management accountant should be defined through an editor or through data conversion. This is mapped to the information found in the BPMS, specifically the executable process model and the organizational model used in the BPMS. All workflow elements in cost drivers are mapped to workflow elements in the process model and organizational model. The mapping is defined as being non-case sensitive. Creation of the editor to define cost drivers and functions and an editor to map cost drivers to the process model and organizational model are out of the scope of this work.

4.2 Log Annotation Implementation

In Section 3.3, the cost model used for event log annotation and the XES extension for the cost annotation of the event log were discussed, as well as the test case example of the cost model. However, how the cost annotation is done is not yet clear and will be further explained.

The input for cost annotation is an event log together with a cost model. The cost model is defined prior to process execution and contains all costs associated with the process in the form of cost drivers. Each cost driver can be applicable for each task instance. The annotation is given in pseudo code in Listing 4.1 and will be further explained.

The event log is annotated in a sequential way. Task instances are composed of several events whereby each event is associated with a single transition in the lifecycle described in Section 2.3.2. Upon task instance completion, sufficient information about the task instance is available to annotate the task instance with cost information. Cost annotation is done in the final event of the task instance. This includes events of type autoskip, manualskip, complete for a successful completion of the task instance. Or withdraw, pi_abort (process instance abort), ate_abort (activity execution abort) for cancellation of the task instance. All events are considered but only final events of a task instance are considered for annotating with the cost of the whole task instance.

For example, a task instance can have the following sequence of events: schedule, assign, reassign, reassign, start, suspend, resume, complete. In this case a number of costs specified in different cost drivers can be associated with the task instance

- Fixed cost associated with the successful completion of the task.
- Variable cost associated with durations of the task.
- Variable cost associated with durations of resource involvement in task execution.
- Variable cost associated with the case data variables and/or values.
foreach Trace in the Event Log do
  foreach Event in the Trace do
    if Event is end event then
      initialize Event Cost per Cost Type;
      foreach Driver do
        if Driver is applicable then
          calculate cost from Driver for Event;
          if Cost is variable due to event Data then
            calculate adjusted cost for event data;
          end
          if Cost is variable due to Event duration then
            calculate adjusted cost duration;
          end
        end
      end
      add to Event Cost per Cost Type from Driver Cost Type;
    end
  end
  annotate Event with Event Cost per Cost Type;
end

Listing 4.1: Event log cost annotation algorithm

All costs will be stored per cost type in the final event of the task instance having lifecycle state complete. Sections 4.2.1 and 4.2.2 explain in more detail how cost itself is calculated. For each final event, all cost drivers are considered and checked is whether each cost driver is applicable for the cost annotation of the event depending on the workflow elements the cost driver is associated with. The cost driver is selected for the cost annotation of the event if all of the following conditions are satisfied.

- If the cost driver is associated with a task, the event has to be associated with the same task.
- If the cost driver is associated with resources, the event has to be associated with at least the same set of resources.
- If the cost driver is associated with data variables, the event has to be associated with at least the same set of data variables.
- If the cost driver is associated with data variables having values, the event has to be associated with the same set of data variables having the same values.

If the cost driver is applicable for the event, the associated cost together with its value and currency are taken as the basis for variable cost calculation.
4.2.1 Variable Cost Depending on Data

The cost amount of the cost driver is implemented as an expression where variables should match the names of case data variables included in the cost driver. Case data variables can have an optional value which is used in cost driver selection described in Section 4.2. However, for the purpose of variable cost calculation depending on case data, the case data variables should not have a value since the case data variable is used as an actual variable in the cost driver amount expression.

As an example, Section C.2 contains a cost driver from the test case example specifying variable cost depending on case data. Whenever the task “Analyze Defect” is executed in the example, a defect type is assigned to the case as a case data variable specifying the severity of the defect. In the example, minor to severe defects are rated on a scale of 1 to 10. The amount of the cost in the cost driver is a formula, “1 * defectType”, and is based on the severity of the defect which is 1 AUD per point on the scale. The cost will be from 1 to 10.

It should be noted the expression evaluation assumes case data variables of integer or floating point. However, improvement of the math expression evaluator to include for example boolean values determining alternate values for the cost of the cost driver is recommended.

4.2.2 Variable Cost Depending on Duration

Another type of variable costs is derived from durations of task instances. Six types of durations have been defined. Three types of durations are associated with the task instance itself and three are associated with the resources taking part in the task instance execution. How the durations correspond to the states in the lifecycle (in Figure 2.4) is highlighted in Figure 4.1 (adapted from [13]).

In the figure the association of states with durations is determined by which costs associated with durations are of interest for cost reporting. There are two durations associated with the state between the “reassign” and “start” transitions. These durations, “Resource Assigned” and “Resource Allocated” make the distinction between the situation where a resource is assigned to a task instance to which later another resource is assigned, or the situation where the assigned resource eventually started working on the task instance, respectively.

The six types of durations are captured in the cost driver together with the unit type of the durations. In the current implementation, the unit type for durations is either hour or minute. If the duration type is specified in the cost driver and is one of the activity durations types, there should be a task associated with the cost driver. Similarly, if the duration type is specified in the cost driver and is one of the resource durations types there should be a resource associated with the cost driver. The actual durations are calculated from the events in the task instance.

An example of a cost driver which defines the cost associated with the duration of resource involvement in a task instance is shown in Appendix C.3. The cost of resource “Tester1” being busy with a task instance of task “Analyze Defect” is 20 AUD per hour. It should be noted that in the test case example only the busy or working duration types are present. However, the implementation covers all six duration types.
In case the cost driver specifies both case data and a duration, first the variable cost depending on case data is calculated, and this result is taken as the input for variable cost calculation depending on the duration.

4.2.3 Log Event Annotation Restrictions

Cost drivers are designed to be associated with multiple cost types and multiple resources. However, the flexibility of XES event log extensibility imposes some restrictions on log event annotation with cost information.

Annotating log events with cost assumes the currency to be linked to the most fine-grained cost information. Due to restrictions in the current event log annotation format, however, the currency association is made at the level of an end event itself, while multiple types of cost through multiple cost drivers can be associated with an end event. Since nesting information in annotations of event logs prohibits use of the annotated log by a multitude of mining and analysis plug-ins, the currency of the last applicable cost driver found in the algorithm is used to associate with the end event.

The organizational extension of XES events logs assumes a single resource to be associated with a task instance at any point in time [13]. This problem is addressed in other work, where richer resources to task associations are suggested [19]. Since this work uses XES event logs for cost annotation, only one resource (identified directly or by its role or group) is assumed to be working on a task instance at any point in time.
4.3 Reporting Implementation

Cost annotated event logs contain fine-grained cost information and can be used to generate cost reports aggregating cost by case, cost type, task, case data, or any other data available in completed task instances. Cost functions have been defined in the cost model defining the content of cost report. Dynamic reporting using cost functions is left as future work. Static reports are implemented taking in a cost annotated event log producing an overview of cost by case, cost type, or both, showing the potential for using cost annotated event logs for cost reporting.

The cost annotated event log with data from the test case example is used to generate a cost report with cost aggregated per case and cost type, which is shown in Figure 4.2. (The report contains 1000 cases and is too large to be included as a whole.) Cases are identified by a “Trace ID” in the cost report. Each row shows cost associated with a single case, every column shows the cost for a cost type.

![Table: Cost for trace and cost type in currency AUD](image)

The cost report results are validated by example using case two. For this case, interestingly, repairs have failed the maximum of five times incurring a cost of 200,- AUD for cost type “Failed Repair”. (Which is explained as the cost of a replacement phone for the customer.) Using Table 2.3, showing the static cost associated with the phone repair process and through direct inspection of the event log, the total cost for case two in the cost report is validated as follows.

![Figure 4.2: Cost per case and cost type](image)
From inspection of the event log, it is noted case two handles a phone of type “T2”, with defect type 4. Furthermore, defect analysis takes 8 minutes, total repair time for five simple repairs is 69 minutes, and the total testing time for testing five repairs is 38 minutes. The calculation using the values from the event log for case two and the static cost associated with the phone repair process is shown in Table 4.1. The cost per cost type at well as the total cost for case two match the values in the cost report. (The one cent difference stems from intermediate rounding of cost per cost type. In the cost report rounding is done just before display of a cost figure and is most accurate.)

<table>
<thead>
<tr>
<th>Cost Type</th>
<th>Description</th>
<th>Calculation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administration</td>
<td>Fixed cost: 5 failed repairs * 10</td>
<td>5 * 10</td>
<td>50.00</td>
</tr>
<tr>
<td>Analysis Complexity</td>
<td>Variable cost: defectType * 1 + defectType * minutes * 0.01</td>
<td>4 * 1 + 4 * 8 * 0.01</td>
<td>4.32</td>
</tr>
<tr>
<td>Facilities</td>
<td>Hourly cost: defect analysis *10 + repair testing * 15</td>
<td>8 / 60 * 10 + 38 / 60 * 15</td>
<td>10.83</td>
</tr>
<tr>
<td>Failed Repair</td>
<td>Archiving of a failed repair and replacement phone: 200</td>
<td>1 * 200</td>
<td>200.00</td>
</tr>
<tr>
<td>Parts T3</td>
<td>Fixed analysis cost for a phone of type &quot;T3&quot;: 20</td>
<td>0 * 20</td>
<td>0.00</td>
</tr>
<tr>
<td>Wages Solver</td>
<td>Hourly wage solver: 65</td>
<td>69 / 60 * 65</td>
<td>74.75</td>
</tr>
<tr>
<td>Wages Tester</td>
<td>Hourly wage tester: defect analysis * 60 + testing repair * 70</td>
<td>8 / 60 * 60 + 38 / 60 * 70</td>
<td>52.33</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>392.23</td>
</tr>
</tbody>
</table>

Table 4.1: Costs per cost type calculation for case two

The cost report generation is implemented as a plug-in directly in the implementation environment. The presentation of the cost report through a presentation library, and exporting of cost reports are implemented in this work. Appendix D contains two other examples of cost reports showing the cost per case and the cost per cost type. Presentation of cost reports using report templates and generic reporting using cost functions are future work.
Chapter 5

Conclusions

The incentive for this work comes from problems encountered in management accounting. This discipline and research field explores ways to capture the real cost of operations accurately and process this fine-grained cost information into a form suitable of decision making. Early costing technique ABC proved a sound way to capture cost accurately but suffered from its substantial effort to implement and to keep up to date. Two successors, TDABC and RCA, refined and improved ABC principles and require less effort to implement and to keep up to date. However, information gathering for the purpose of cost reporting and decision making remains a significant part of management accounting and can be costly in itself.

Through BPM and the use of information available in a PAIS the information gathering effort of a management accountant can be reduced or even automated. Since a PAIS contains fine-grained information about the process in the form of event logs, associating cost with this information gives an accurate insight into the cost of a process, and can be used for reporting and decision making. This has been captured in a goal and sub goals in the introduction which are repeated below.

Goal summary: Design an architecture to make process mining cost-aware, thereby supporting the information gathering phase and decision making phase in management accounting.

Sub goal 1: Define in a flexible format aspects which drive cost and aspects required for cost reporting and link these to concepts found in BPM.

Sub goal 2: Find a structural way to incorporate cost information in event logs from a PAIS.

Sub goal 3: Create an implementation using the flexible format to enhance event logs with cost information.

Sub goal 4: Generate reports from the event logs to support decision making in management accounting.

In this conclusion the measure of fulfillment of the goal and sub goals are discussed as well as limitations, and recommendation for future work.

Central to management accounting is assigning cost to resources, activities and products or services through cost drivers. In Chapter 3 this concept is linked to the resource, task and case data dimensions grouped together as workflow elements. The allocation of cost to workflow elements is kept flexible putting few restrictions on ways cost can be incurred. Variable cost is related to either durations of tasks,
durations of resources working on tasks or case data. The definition of cost drivers in a workflow context allows for the assignment of cost to historical information in an event log. The structure of the cost annotated event log is modeled. Reporting of cost is supported through cost functions defining the contents of a report. Together, this analysis of required information covers the first and second sub goals.

The realization of the third sub goal is described in Chapter 4. Conceptual models have been translated into data models and are used as informational input for annotating an event log with cost. Each completed task in the event log can be annotated with cost information through zero or more cost drivers. Applicability of cost drivers for completed tasks is determined through the resources, tasks and case data variables defined in both the event log and the cost driver. The cost of a completed task in the event log is fixed, variable or a combination defined through multiple cost drivers. Variable cost is defined in the cost driver through duration of a task, duration of a resource working on task or case data.

The fourth sub goal has been realized partially. Dynamic reporting using cost functions to generate reports has not been implemented. Static reporting has been implemented which aggregates cost from a cost annotated event log in various ways and presents this cost in a tabular form. From this it is concluded the costing techniques mentioned in this work are now supported though process mining. The main contribution of this work and fulfillment of its main goal is a proof of concept implementation to support this conclusion.

The deliverables making support of costing techniques though process mining possible are listed in Section 5.1. Limitations and refinements that should be implemented are mentioned in Section 5.2.

5.1 Deliverables

The design and implementation consist of a number of artifacts which are categorized as conceptual models, data formats and algorithm implementations and are listed below.

- **Cost model** A conceptual data model of cost elements to be included in this work represented in the ORM 2 notation.

- **Log annotation model** A conceptual data model for event log cost annotation in ORM 2 notation.

- **Cost model data definition** An XML schema definition formalizing the conceptual cost model to be used in the implementation environment.

- **Log annotation data definition** An XESEXT data definition for event log cost annotation.

- **Log annotation algorithm** Algorithms for the serialization of cost models defined in the cost data model format, and an algorithm for log annotation based on cost drivers and an event log. This last algorithm is extended with duration calculation based on sequences of events in the event log and which is needed for log annotation.
Cost reporting algorithms Cost reporting based on cost annotated event logs.

Furthermore, an instance of the cost model XML schema definition has been created for the phone repair process test case example. Through the described algorithms, this cost model instance together with an event log from the test case example were used to generate a cost annotated event log. This cost annotated event log conforms to the XES event log format and the cost annotation data definition.

5.2 Limitations and Future Work

In this work input and output data formats were created, namely the cost model as an XML schema definition and event log extension data definition in XESEXT format. One assumption was that XES logs would be produced by a PAIS through execution of a process. Another assumption was that a management accountant would have obtained data from ERP, HRM, CRM and/or other systems and had defined costs in a cost model in the desired cost model XML schema definition. In many organizations it is unrealistic to expect an event log and cost model to be available in their respective desired formats. The necessity is recognized to identify relevant data sources and convert data from these sources to the desired data formats. Hence, information gathering and conversion needs to be done before the results of this work can be applied. In Section 2.3.1 conversion tools for XES are mentioned. Converting gathered information to the desired cost model data format is future work.

Extensions to XES event logs are defined in the XESEXT format and cost annotation of XES event logs has therefore been defined in this format. While implementing the event log cost annotation algorithm, it was found the nesting of information in the events was undesirable due to the fact that many plug-ins in the implementation environment expect non-nested information. Hence, in an event, a single resource and currency are defined despite cost drivers possibly having multiple resources and possibly using different currencies. Support for nesting in the structure of XES events could also benefit work exploring richer resource models [19]. Currently, the XES extension for resources supports one resource per event which can be an individual resource, a role, or a group [13].

The current implementation can be improved in a number of small ways. The XES cost extension was defined in such a way that events can be annotated with multiple cost types. Each cost driver can also have multiple cost types. During implementation it was found that because of this, cost can be incurred more than once in a single event. That is, once for each cost type associated with a cost driver. Hence, in the test case example referred to throughout this work, each cost driver has exactly one cost type. An intuitive way to support multiple cost types and split total calculated cost driver costs out over the cost types is future work.

Table 2.2 shows case data variables used in the phone repair process and the tasks setting these variables. In the current implementation, data values have to be present in a final event of a task instance in order for a cost driver having that same data variable to be applicable for annotating the event. If the case data variable is set during a task execution earlier in the execution of the case and is therefore part
of the case, the current implementation does not recognize the case data variable and its value when annotating events later in the case execution. For example, the case data variables “phoneType” and “defectType” are set during execution of the “Analyze Defect” task but their values cannot be used for cost annotation of later events like “Test Repair”.

In the current implementation data values for case data variables used in cost calculations are assumed to be from a numerical domain for evaluation of the cost expression in a cost driver. Supporting domains like booleans or enumerations in the evaluation of the cost expression can also be conceived of.

Although static cost reporting has been implemented, dynamic cost reporting through cost functions is desirable and left as future work. In Section 1.3 visualization and prediction are mentioned. Besides reporting, visualization and prediction are useful for decision making. This work is likely to benefit from advances in visualization and prediction in process mining.
Bibliography


38


Appendix A

Cost Model

A.1 Cost Input

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="costInput">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="drivers" minOccurs="0"/>
        <xs:element ref="functions" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="drivers">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="driver" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="metadata">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="metadata"/>
        <xs:element ref="workflowElements"/>
        <xs:element ref="costTypes"/>
        <xs:element ref="unitCost"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="workflowElements">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="workflowElements"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="costTypes">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="costTypes"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="unitCost">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="unitCost"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```
<xs:element name="name" type="xs:string"/>
<xs:element name="description" type="xs:string"/>
<xs:element name="type" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="workflowElements">
<xs:complexType>
<xs:sequence>
<xs:element name="workflowElement" maxOccurs="unbounded">
<xs:complexType>
<xs:attribute name="id" type="xs:ID"/>
<xs:attribute name="type" use="required">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="task"/>
<xs:enumeration value="resource"/>
<xs:enumeration value="data"/>
<xs:enumeration value="unknown"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="name" type="xs:string" use="required"/>
<xs:attribute name="value" type="xs:string"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="costTypes">
<xs:complexType>
<xs:sequence>
<xs:element name="costType" type="xs:string" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="unitCost">
<xs:complexType>
<xs:sequence>
<xs:element name="amount" type="xs:string"/>
<xs:element name="currency" type="xs:string"/>
<xs:element name="unit"/>
<xs:simpleType>
<xs:restriction base="xs:string">
  <xs:enumeration value="minute"/>
  <xs:enumeration value="hour"/>
  <xs:enumeration value="invocation"/>
  <xs:enumeration value="unknown"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="duration">
<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:enumeration value="suspended"/>
    <xs:enumeration value="waiting"/>
    <xs:enumeration value="busy"/>
    <xs:enumeration value="assigned"/>
    <xs:enumeration value="allocated"/>
    <xs:enumeration value="working"/>
    <xs:enumeration value="unknown"/>
  </xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="functions">
<xs:complexType>
<xs:sequence>
  <xs:element name="function" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="description" type="xs:string"/>
        <xs:element name="returnType" type="xs:string"/>
        <xs:element ref="parameters"/>
        <xs:element name="expression" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="parameters"/>

42
Listing A.1: Cost input

A.2 Cost Model

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xsi="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="cost_input.xsd"/>
  <xs:element name="costModel">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="mappings" minOccurs="0"/>
        <xs:element ref="drivers" minOccurs="0"/>
        <xs:element ref="functions" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="mappings">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="mapping" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="mapping">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="identifier" minOccurs="2" maxOccurs="2"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

43
Listing A.2: Cost model
Appendix B

XES Cost Extension

```xml
<?xml version="1.0" encoding="utf-8"?>
<xesextension name="Cost" prefix="cost" uri="http://www.xes-standard.org/cost.xesext">
  <log>
    <string key="currency">
      <alias mapping="EN" name="currency"/>
    </string>
  </log>
  <trace>
    <string key="currency">
      <alias mapping="EN" name="currency"/>
    </string>
    <float key="type">
      <alias mapping="EN" name="cost type and amount"/>
    </float>
  </trace>
  <event>
    <string key="currency">
      <alias mapping="EN" name="currency"/>
    </string>
    <float key="type">
      <alias mapping="EN" name="cost type and amount"/>
    </float>
  </event>
</xesextension>
```

Listing B.1: XES cost extension
Appendix C

Case Example

C.1 Mapping

```xml
<mapping type="resource">
  <identifier type="cost" id="ea542ef2-0738-4c46-ab75-0acce298e8b" name="Tester1"/>
  <identifier type="workflow" id="bfe47e59-2a78-452b-b818-d43915558441" name="Tester1"/>
</mapping>
```

Listing C.1: Mapping

C.2 Cost Driver with Variable Cost from Data

```xml
<driver id="0a959610-047e-414c-b245-0f520d01cbe5">
  <metadata>
    <name>Analysis Complexity</name>
    <description/> 
    <type/>
  </metadata>
  <workflowElements>
    <workflowElement id="642bdc62-4b53-49f8-8e8f-6c60e101022c" type="task" name="Analyze Defect"/>
    <workflowElement id="f30cb896-44c4-41a9-bfbe-ab335bf76f44" type="data" name="defectType"/>
  </workflowElements>
  <costTypes>
    <costType>Analysis Complexity</costType>
  </costTypes>
  <unitCost>
    <amount>1 * defectType</amount>
    <currency>AUD</currency>
    <unit>invocation</unit>
    <duration>unknown</duration>
  </unitCost>
```

46
Listing C.2: Cost driver with data

C.3 Cost Driver with Variable Cost from Duration

Listing C.3: Cost driver with duration

C.4 Log Event Annotation
Listing C.4: Event log cost annotation for event
Appendix D

Cost Reports

D.1 Cost per Case

Figure D.1: Cost per case
D.2 Cost per Cost Type

<table>
<thead>
<tr>
<th>Cost Type</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administration</td>
<td>12540.00</td>
</tr>
<tr>
<td>Analysis Complexity</td>
<td>6072.93</td>
</tr>
<tr>
<td>Facilities</td>
<td>5278.33</td>
</tr>
<tr>
<td>Failed Repair</td>
<td>17600.00</td>
</tr>
<tr>
<td>Parts T3</td>
<td>6960.00</td>
</tr>
<tr>
<td>Wages Solver</td>
<td>41735.25</td>
</tr>
<tr>
<td>Wages Tester</td>
<td>26288.00</td>
</tr>
</tbody>
</table>

Figure D.2: Cost per cost type