UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering,
Mathematics & Computer Science

Enterprise Architecture Mining

Ahmad Mujahid Fajri

Master Thesis
February 2019

Study Programmes
MSc Computer Science (CSC)
MSc Business Information Technology (BIT)

Supervisors
Prof. dr. Maria-Eugenia Iacob
dr. ir. Marten van Sinderen

University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Abstract

In order to maintain its competitive advantage, an enterprise needs to be adapted to changes and opportunities. EA is one of the tools that capable to grasp the current condition of the enterprise. Thus it is prevalent to maintain an up-to-date EA model. However, manually maintain the model is cost and time consuming. In order to automated maintenance process, there is available method called automated EA model documentation. They are tools, mechanisms that enabled an architect to maintain the EA model automatically. However, current tools and methods only limited to certain systems or products. In this research, we propose an alternative to conducting automated EA model documentation that can combine multiple data sources and inter-operable structure between systems.

The research conducted a literature review to study current literature related to log, event log, types of the log, and how an event log produced based on the viewpoint of process mining. The literature study also discovers the definition of process mining, its categories, its type of perspective and the algorithms that support the mining process. The study also discovers possible data sources that available for automated EA model documentation, related work in the automated EA model documentation and lastly propose the conversion pattern between a process model and an EA model. The research also did an narrative review to select appropriate process mining algorithms that are needed for the validation process.

We also proposed a log structure that can be populated from systems with the help of the log guideline. Moreover, we indicate possible relevant fields that can be added to the structure to gather additional EA elements. After that we propose EA mining that consists of three steps, to discover business process, elements that related to the workflows, and analysis function. Using both log structure and EA mining we were able to generate an EA model. We were also able to implement EA mining and create algorithms and a prototype. In the research, we also conducted validation to test both the structure and the EA mining. The validation is analysed if the user perception comply with the reality that are produced from running systems. The validation also consists of the accuracy of the conversion pattern and the performance of the prototype.

Preface

This thesis is a requirement that is needed to get a master degree in Business Infomation Technology at the University of Twente. In the past two years, I gained a lot from this programme. I received new information and knowledge, experiencing different cultures, working style, and meeting with new people.

Thank you for Allah SWT for this opportunity and experience, for his providence and guidance during my study. I also would like to express my gratitude to the Ministry of Communication and Information (MCIT) of the Republic of Indonesia. Without the scholarship that was granted to me in 2017, it will be hard for me to get this opportunity. And it has always been an honor for me to be an MCIT scholarship awardee.

I would like to thank my family that always supports me through my study. My wife Citta, both of my daughters: Kaffa and Alisya. You guys have always helped me in my dire time and also motivate and cheer me up through that time. I would like to dedicate this thesis to my parents: Mama Bibah and Ayah David. Without your support and prayers, I wouldn't be the person that I am today, and I wouldn't be to where I am now. My parents-in-law that always support me. Thank you for Bapak Sobrun that always visited and looked after my family in my absence and Mama Ely for supporting me financially.

Thank you for my supervisors: Marten, Maria, and Adina. Without your guidance and support, i will not be able to finish my thesis and my degree here. I would like to thank all my professors and lecturers that share their abundance knowledge with me. I am hoping that we can meet again in the next occasion, and the knowledge that you bestow upon me hopefully can always help me through my journey in the future.

Thank you for my Indonesian friends and families in Enschede and the Netherlands, thank you for your friendship, help, and moments that you guys share with me. And to other people that I cannot mention one by one, thank you for being a part of my journey during my study. I wish you all the best, and I hope we will meet again in the future.

Ahmad Mujahid Fajri
Enschede, 26 February 2019

Contents

List of Figures

List of Tables

1 Introduction

1.1. Motivation

Frequent changes in socio-economic environments are continuously challenging enterprises. These changes can be varied, from rapid transitions in business models, compliance with new regulations, or introduction of new business services and technologies. In order to navigate through those changes, an enterprise needs a guideline, a tool that enables the enterprise to see its own capabilities in business and information technology. Enterprise Architecture (EA) could provide assistance for the enterprise in designing and realising of the enterprise's organisational structure, business processes, information system, and infrastructure [28]. It could also give a holistic overview of the enterprise and providing necessary information for decision-makers.

Currently, enterprises are struggling to maintain up-to-date EA models. The survey conducted by Winter et al. [46] stated that EA models maintenance process still conducted in a highly manual process with little automation. Moreover, the maintenance process can not cope up with the growth of the enterprise, and it leaves the models became (partly) outdated [5]. In addition, the EA delivery function could also suffer from ivory tower syndrome [41], which leads to deliver EA models with wrong level abstraction, that might be too abstract or complex to be used in practice. Combination of manual processes with a high volume of changes that are needed to be maintained, and sometimes the reality is quite differs from what architects perceived leads to maintaining EA models are time consuming and costly.

Some attempts were made by researchers in tackling manual maintenance processes by introducing automated EA documentation. Farwick et al. [14] and Valja et al. [35] conducted research of requirements for maintaining an automated EA model, Hauder et al. [21] studied challenges in the maintenance process. In addition, Holm et al. [22] studied the usage of a network scanner for automatic data gathering process to create an EA model. Farwick et al. [12] presented semi-automated processes for EA data collection and quality assurance, they also made an extension of EA maintenance processes to meet the requirements for EA automated maintenance. Furthermore, they argued that the requirements could be a basis for future technical implementation. Buschle et al. [9] utilised an Enterprise Service Bus (ESB) to automate an EA documentation. They reverse-engineered the ESB data model and made transformation rules for three layers of an EA framework. They argued that automated processes could reduce cost and data quality improvement. Johnson et al. [25] described the usage of Dynamic Bayesian Networks (DBNs) for automatic EA modelling. They argued that this approach could help in automating the modelling processes. Van Langerak et al. [42] studied the utilisation of process mining in uncovering cooperation of each department of an organisation by analysing execution data. They define a social network analysis of the organisation using a log that is produced by running systems. In this study, they implemented an automation process in data gathering by tapping information on the running systems and create new Archimate viewpoint as an output.

There is also other research that specified in creating an automated EA model through

using a log [42] or other data sources([9], [22]). However, there are limitations in their research. Mostly lies in the tools that were used. In [9] they used SAP PI as an ESB. In the tool not all information is available to generate an EA model, as the tool is technology-oriented, hence lacking in providing some business perspective. While [42] limiting their research to certain aspects of viewpoint (Business Process Collaboration), and the technique implies that it still required manual processing as they used Process Mining to generate Process Model and Social Network Analysis before converting them into an EA model. While [22] has similar circumstances as in [9], they dependent on the tools. Since the data might or might not available for conversion, thus, limiting the model that was produced.

1.2. Research Design

Research design consists of a research goal, research methodology, and thesis structure. Research goal will discuss the objective of this research and formulate it into research questions. While research methodology explains that methods that were used to answer the research questions. Lastly, the thesis structure explains the writing structure of the thesis, what to expect for each chapter of this thesis, and its alignment with research questions and research methodology.

1.2.1 Research Goal

The main objective of this research is to produce artefacts that can convert a daily log activity into an EA model, and the objective of the artefact is to promote an alternative to approaches that are currently available in automating EA model documentation. In addition, processing daily log activity can close the gap between the user's perception with reality. This research has a main research question of ***how to convert a log into an EA model?***. The main research question is supported by two sub-questions, the sub-questions talk about two big parts of the research: the input and the process. Each sub-questions is supported by additional supporting questions. The structure of research questions can be seen at Fig.1.1.

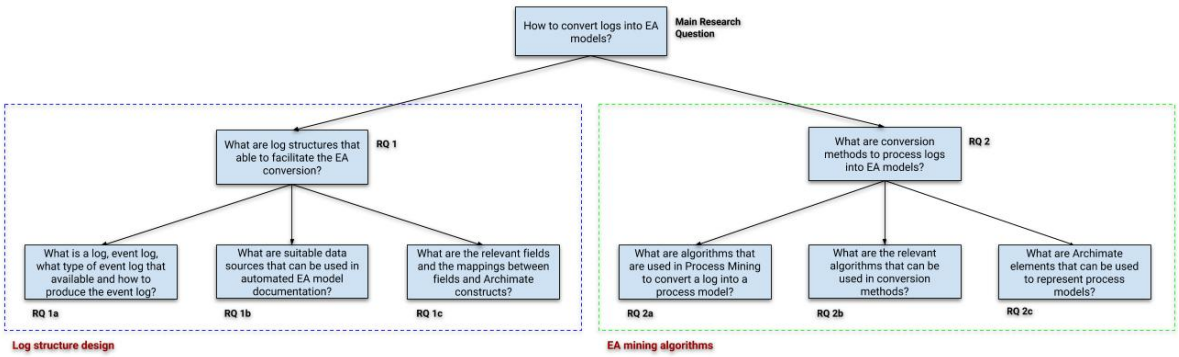


Figure 1.1: Research questions overview

RQ1. What log structures that are able to facilitate the EA conversion?

The input is needed for the EA conversion, and what structure that can be accepted into the conversion mechanism? In order to answer this question, there are sub-questions that are needed to be answered first. Detail of the sub-questions can be seen at the following list. The research conducted a systematic literature review with exploratory literature review and synthesis the answer to this question. The result can be seen at Section.4.1.1.

RQ1a. What is a log, event log, what type of event log that available and how to produce the event log?

The conversion mechanism is needed logs as input. Hence it is important to understand the definition of a log, event log, types of the event log that available currently

and how to produce the event log. The answer to this question can be seen in Section.2.2.1.

RQ1b. What are suitable data sources that can be used in automated EA model documentation?

An event log can be produced by multiple data sources and what are suitable data sources that available as an input for the log. The research conducted a systematic literature review to discover suitable data sources that are available for automated EA model documentation, and the result can be seen at Section.2.2.2.1.

RQ1c. What are the relevant fields and the mappings between fields and Archimate constructs?

After knowing the suitable data source, then the next questions is which are the relevant fields that can be used? and what is the mapping between those fields to Archimate constructs. This research conducted a systematic literature review to answer this question, and it can be seen at Section.2.2.2.2

RQ2. What are conversion methods to process logs into EA models?

After knowing the input, the next question of this research will be how to process that input to be the expected result? Next, it is necessary to answer the sub-questions first. The sub-questions can be seen at the following list. After answering the sub-questions, the research conducted a treatment design and created the conversion definition (Section.4.2.1), implementation (Section.5.1), and prototype (Section.5.3).

RQ2a. What are algorithms that are used in Process Mining to convert a log into a process model?

The conversion methods were derived from process mining, then it is important to know what is process mining, a different perspective of process mining, and algorithms for each perspective.Next, the research conducted a systematic literature review and able to produce a list of suitable algorithms and miner that can be used for the purpose of the conversion, the result can be seen at Section.2.2.3.2.

RQ2b. What are the relevant algorithms that can be used in conversion methods?

After knowing the algorithm and the miner that can be used for the conversion methods, the next step of the research is to pick the relevant algorithms that suitable for the conversion. Next, the research conducted a narrative review, and the result can be seen at Section.3.2.2.

RQ2c. What are the Archimate elements that can be used to represent process models?

This study used relevant algorithms from process mining, the algorithm then are incorporated into EA mining, and the process model that the algorithm produced needs to be converted into Archimate elements. This research used metonymy to associate the elements of the process model to elements of Archimate. The result of this process can be seen at Section.2.2.4.

1.2.2 Research Methodology

This research used the Design Science Methodology (DSM) [45]. The design science is a suitable framework to investigate and design an information system (IS) artefact. It is also defined interactions between artefact and the problem context in order to make improvements in the context. The DSM introduces the design cycle to iterates over the activities of designing and investigation of a design science research project. The design cycle consists of three tasks: The *problem investigation* is to examine problems that will be addressed by artefact using context observation. Finding the causes, mechanisms, and reasons behind those problems. The *treatment design* is to specify requirements for the artefact, correlate the requirements to research goals, and, designing treatments to address the problems. Lastly, the *treatment*

validation is to examine the satisfaction level between the artefact and the research objectives.

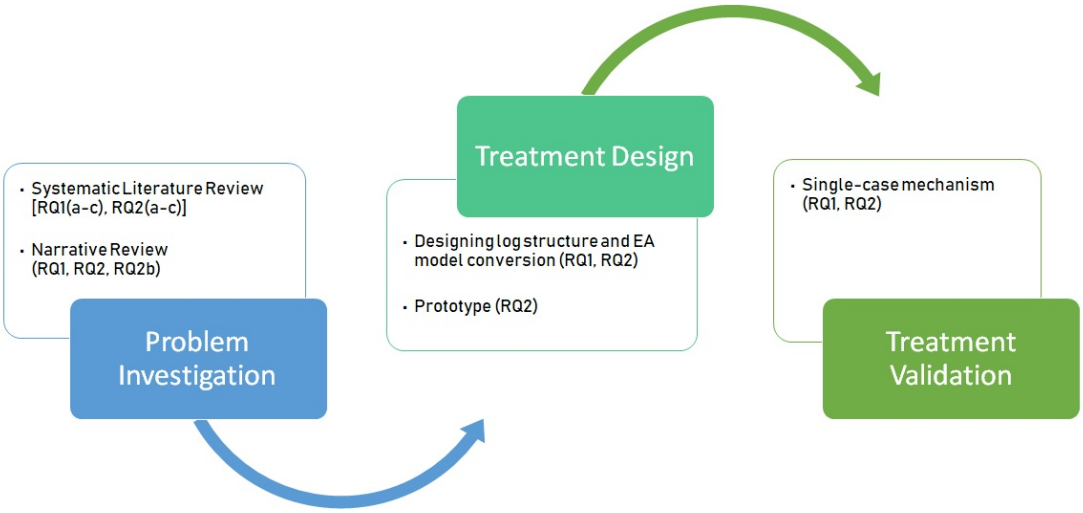


Figure 1.2: Research design strategy

This research used various research approaches. Each approach was associated with a step in DSM, and each step of DSM was used to answer a specific research question. The association between approach, DSM step, and research questions can be seen in Figure.1.2. The list of the approaches can be seen at the following list:

Systematic Literature Review

This research used systematic literature review (SLR) [27]. An SLR is a methodologically rigorous review of research results. The objective of an SLR is to support the development of evidence-based guidelines for the practitioner and to aggregate all existing evidence on a research question. The SLR helped the research to investigate problems and answering research questions **RQ1(a-c)** and **RQ2(a-c)**.

Narrative Review

Narrative review is a study that focused on gathering relevant information that provides both context and substance to the author's overall argument [47]. This approach complements the SLR to investigate the problem and was used to select suitable algorithms for the conversion methods (**RQ2b**). Besides, this approach also helps to design the artefacts (**RQ1, RQ2**).

Prototype

Prototypes are widely recognised to be a core means of exploring and expressing designs for interactive computer artefacts [23]. Prototypes provide the means for examining design problems and evaluating solutions. This research built the prototype in the treatment design, it helped the research in validating the research's constructs (**RQ2**), and it provided feedbacks for further improvement of the artefacts.

Single-Case Mechanism Experiment

A single-case mechanism experiment (SCME) [45] is a test to describe and explain cause-effect behaviour of the object of study. The research used SCME in the treatment validation, the objective of this test is to obtain the response of the internal mechanism of a validation model if the model were to be tested by certain stimuli. This test helped to validate the conversion methods (**RQ1**) and the log structure (**RQ2**).

1.2.3 Thesis Structure

This paper is structured as follows: Chapter two presents a systematic literature review (SLR). The chapter discusses: searching methodology, findings, and discussion. This chapter will be answering **RQ1(a-c)**, **RQ2(a-c)**. Chapter three presents the theoretical background, this chapter will answer **RQ1** and **RQ2**. The theoretical background also adds the additional theory that needed for this research. Chapter four describes the artefact for this research, it will answer **RQ1** and **RQ2**. Chapter five is the implementation of the EA mining, and it also produces a prototype. This chapter will answer **RQ2**. Next, chapter six presents the validation of this research. The validation methods and results. this chapter will answer **RQ1** and **RQ2**. After that, chapter seven discusses the result of this research, concludes the report, and provides suggestions for future work. The following table gives an overview of the research structure and the traceability matrix between chapters, DSM phases, and research questions.

Table 1.1: Thesis structure and tracebility matrix

Chapter	Applicable DSM phases	Research Questions
1. Introduction	-	-
2. Literature Review	Problem Investigation	RQ1(a-c), RQ2(a-c)
3. Theoretical Background	Problem Investigation	RQ1, RQ2
4. EA mining	Treatment Design	RQ1, RQ2
5. Implementation	Treatment Design	RQ2
6. Validation	Treatment Validation	RQ1, RQ2
7. Discussion, Conclusion and Future Works	*All DRM phases*	*All research questions*

2

Literature Study

In this chapter we will discuss the literature study that we conducted in the problem investigation phase, in order to extract information regarding event log, available data sources for automated EA model documentation, algorithms that were used in process mining to produce process models, and lastly conversion pattern that we used for associating a process model to an EA model.

2.1. Literature Review Methodology

In this research we conducted a systematic literature review (SLR) using Kitchenham et al. [27] framework, each steps in SLR method are described in detail in the following subsections.

2.1.1 Search process

In search process we first looked into Scopus and Web of Science for preliminary search for title and abstract. After that, we looked into other digital libraries to get the full text of the literature. We also conducted backward and forward search for literature that we find useful but not yet covered in the initial search. Fig.2.1 depicts the outline of our search strategy.

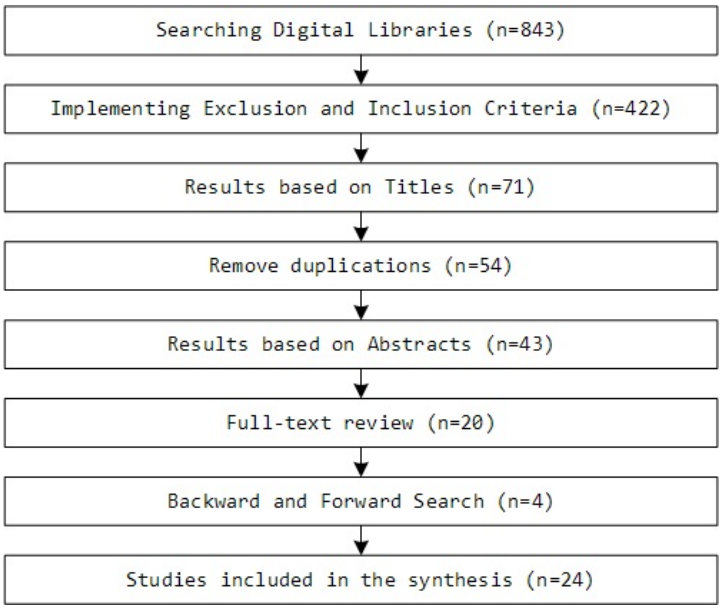


Figure 2.1: Search strategy diagram

We used following keywords to find relevant studies for our research: (”logging” AND ”log management”), (”logging” AND ”literature review”), (”process mining algorithm” AND ”literature review”), (”process mining algorithm” AND ”process discovery”), (”process mining” AND ”control flow perspective”), (”process mining” AND ”organizational perspective”), (”auto*” AND (”enterprise architecture model*” OR ”enterprise architecture documentation” OR ”enterprise architecture”)), (”process mining” AND ”enterprise architect*”) and the following list are the digital libraries that we used in our research.

- Scopus (www.scopus.com).
- Web of Science (www.webofknowledge.com).
- IEEE Explore (www.ieee.org/web/publications/xplore/).
- Research Gate (www.researchgate.net).
- Springer Link (www.springerlink.com).
- Science Direct (www.sciencedirect.com)
- Google Scholar (www.scholar.google.com).
- University of Twente Library (www.utwente.nl/en/lisa/library)

2.1.2 Inclusion and Exclusion Criteria

Inclusion Criteria:

- Studies that related to automated enterprise architecture documentation, process mining and enterprise architecture, process mining algorithm in the literature review, process mining in organisational and control flow perspective, process mining algorithm in process discovery, log management in the literature review, logging and log management.
- Research areas in Computer Science.
- English peer review studies including Conference papers, Proceeding papers, Articles, Books and Book Chapters.
- Published between 2000 and 2018.

Exclusion Criteria:

- Studies are not in English.
- Studies are not related to the research questions.
- Duplicate studies (by title or content).
- Short paper.

2.1.3 Data collection

The data extracted from each study were:

- Identity of study: the Unique identity of the study.
- Bibliographic references: Authors, year of publication, title and source of Publication.
- Type of study: Book, journal paper, conference paper, article.
- Type of Logs: Definition and type of logs.
- Process mining classification: Categorisation of process mining type and perspective.

- Process mining algorithms: Description of algorithms with consideration of the classification.
- Automated EA documentation current studies: Contribution of current literature in automated EA documentation areas.
- Mapping EA Framework: EA framework elements that have already mapped in the current literature.

2.2. Literature Review Result

We began with searching literature in Scopus and Web of Science to get 843 studies related to various topics for this research after we implemented exclusion and inclusion criteria we got 422 studies, and we filtered based on title to get 71 studies. After that, we remove duplication for 54 results, and after we read the abstract, we got 43 studies. Next, after thoroughly reading the content we decided to synthesis 20 studies. In addition to backward and forward search, we decided to add four additional studies. Overall, we got 24 studies for this research. The illustration of the process can be seen at Fig.2.1 and the detail search result corresponding to data collection (2.1.3) can be seen at Table.2.1.

Table 2.1: Literature Review Studies

ID	Author	Date	Topic Type	Topic Areas	Source Type	Cited	Source
S1	Chuvakin et al. [11]	2012	Log	Log management	Book	7	UT Library
S2	Rojas et al. [32]	2016	Process Mining	Literature Review	Conference Paper	70	ScienceDirect
S3	Van Der Aalst W.M.P. [36]	2016	Process Mining	Algorithms	Book	426	UT Library
S4	Akman and Demirörs [6]	2009	Process Mining	Process Discovery Algorithms	Conference Paper	13	IEEE
S5	Weber et al. [43]	2011	Process Mining	Process Discovery Algorithms	Conference Paper	9	IEEE
S6	Van Der Aalst W.M.P. [38]	2013	Process Mining	Process Discovery Algorithms	Conference Paper	23	IEEE
S7	Mans et al. [30]	2008	Process Mining	Control Flow Algorithms	Conference Paper	132	Springer Link
S8	Bozkaya et al. [8]	2009	Process Mining	Control Flow Algorithms	Conference Paper	41	IEEE
S9	Kalenkova et al. [26]	2017	Process Mining	Control Flow Algorithms	Article	8	Springer Link
S10	Van der Aalst et al. [40]	2007	Process Mining	Organisational Algorithms	Article	436	ScienceDirect
S11	Song et al. [33]	2008	Process Mining	Organisational Algorithms	Conference Paper	302	ScienceDirect
S12	Appice et al. [7]	2016	Process Mining	Organisational Algorithms	Conference Paper	3	Springer Link
S13	Lismont et al. [29]	2016	Process Mining	Organisational Algorithms	Article	6	ScienceDirect
S14	Aier et al. [5]	2009	EA management	EA maintenance	Conference Paper	38	Google Scholar
S15	Winter et al. [46]	2010	EA management	EA management practice	Conference Paper	83	Research Gate
S16	Farwick et al. [14]	2011	Automated EA	Requirements and Challenges	Proceedings Paper	4	Research Gate
S17	Hauder et al. [21]	2012	Automated EA	Requirements and Challenges	Conference Paper	17	Springer Link
S18	Välja et al. [35]	2015	Automated EA	Requirements and Challenges	Conference Paper	3	IEEE
S19	Farwick et al. [12]	2011	Automated EA	Implementation	Conference Paper	26	IEEE
S20	Buschle et al. [9]	2012	Automated EA	Implementation	Conference Paper	28	Research Gate
S21	Buschle et al. [10]	2012	Automated EA	Implementation	Conference Paper	11	Springer Link
S22	Holm et al. [22]	2014	Automated EA	Implementation	Article	15	Springer Link
S23	Van Langer[42]	2017	Automated EA	Implementation	Conference Paper	0	Springer Link
S24	Johnson et al. [25]	2016	Automated EA	Algorithm	Conference Paper	0	IEEE

2.2.1 Log

RQ1a: *What is a log, event log, what type of event log that available and how to produce the log?*

2.2.1.1 Log and event log

A log is what a computer system, device, software, etc. generates in response to some sort of stimuli [11]. For example, login and logout messages in a Unix system, ACL accept and deny messages in the firewall. Log could be classified into the some general categories [11]: *Informational, debug, warning, error,* and *alert. Informational* is a log that describes occurrences of benign activities, *debug* is a log that aids software developer to troubleshoot and identify problems of running code. *Warning* is a log that describes some situations that might be missing or needed for a system. *Error* is a log that describes errors that occurs in various levels in a computer system. Lastly, *alert* is a log that indicates something interesting has happened. Log has typical basic contents [11], which are: *Timestamp, source,* and *data. Timestamp* is the time at which the log message was generated, *source* represents in IP address or hostname, to describe from which the log was generated. And *data* is the content of the log itself, there is no standard format of how data is represented in a log message. It depends on from which system or application that the log was generated.

While Event log is collection of events used as input for process mining. Events do not need to be stored in a separate log file (e.g., events may be scattered over different database

table) [37]. Another definition of log are:

Definition 1 (Event, attribute [36]). *Let $\mathscr{E}$ be the **event universe**, i.e., the set of all possible event identifiers. Event may be characterised by various **attributes**, e.g., an event may have a timestamp, correspond to an activity, is executed by a particular person, has associated costs, etc. Let **AN** be a set of attribute names. For any event $e \in \mathscr{E}$ and name $n \in AN$, $\#_n(e)$ is value of attribute **n** for event **e**. if event does not have an attribute named **n**, then $\#_n(e) = ?$ (null value)*

Definition 2 (Classifier [36]). *For any event $e \in \mathscr{E}$, $\underline{e}$ is the **name** of the event.*

Definition 3 (Case, trace, event log [36]). *Let $\mathscr{C}$ be the **case universe**, i.e., the set of all possible case identifiers. Cases, like events, have attributes. For any case $c \in \mathscr{C}$ and name $n \in AN$: $\#_n(c)$ is the value of attribute n for case c ($\#_n(c) = ?$ if case c has no attribute named n). Each case has a special mandatory attribute, trace $\#_{trace}(c) \in \mathscr{E}^*$ (In assumption $\#_{trace}(c) \neq \langle\rangle$, i.e., traces in a log contain at least one event). $\hat{c} = \#_{trace}(c)$ is a shorthand for referring to the trace of a case.*

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

An event log is a set of cases $L \subseteq \mathscr{C}$ such that each event appears at most once in the entire log, i.e., for any $c_1, c_2 \in L$ such that $c_1 \neq c_2$: $\partial set(\hat{c_1}) \cap \partial set(\hat{c_2}) = \emptyset$

If an event log contains timestamps, then the ordering in a trace should respect these timestamps, i.e., for any $c \in L$, i and j such that $1 \leq i < j \leq |\hat{c}| : \#_{time}(\hat{c(i)}) \leq \#_{time}(\hat{c(j)})$.

Van der Aalst [36] defined an event can be identified with attributes that it has. Example that was described in the Definition.1, an event can be described in *activity* that the event does, at what time that the event does (*timestamp*), who/what execute the event (*resource*) or what cost that the event takes (*cost*). In addition, we can also identify the event by its name ($\underline{e}$). In summary, event log consists of cases or traces, each case consists of events that each event relates to one case, the event has its attributes, and the event could also be ordered by its attributes. Illustration for an event log structure can be seen at Fig.2.2.

Definition 4 (Simple event log [36]). *Let $\mathscr{A}$ be a set activity names. A simple trace σ is a sequence of activities, i.e., $\sigma \in \mathscr{A}^*$. A simple event log L is multi-set of traces over $\mathscr{A}$, i.e., $L \in \mathbb{B}(\mathscr{A}^*)$. And using assumption that each trace contains at least one element, i.e., $\sigma \in L$ implies $\sigma \neq ()$.*

A simple event log is a multi-set of traces over some set of activity names ($\mathscr{A}$). For example, $[(a,b,c,d)^3, (a,c,b,d)^2, (a,e,d)]$ defines a log containing six cases, and in total there are $(3x2) + (2x4) + (1x3) = 23$ events. In this context all cases start with a and end with d. Moreover, since there are no other attributes in the simple event log, i.e., timestamps or resources, thus, cases and events are no longer uniquely identifiable [36].

2.2.1.2 XES Standard

The current de facto standard for event log is XES ([20], [36]), the XES format is the successor of MXML and referring to practical experiences with MXML, XES format is less restrictive and extensible. The format was adopted by the *IEEE Task Force on Process Mining* in September 2010. The basic structure of XES consists of *Log*. It contains all event information that is related to one specific process. A log contains an arbitrary number (may be empty) of *trace* objects, each trace describes the execution of one specific instance, or case, of logged processes. Every trace contains an arbitrary number (may also be empty) of *event* objects. The event represents a single granularity of activity that has been observed in the execution of a process. The log, trace, and event do not contain information, they defined the structure of the document. All information of an event log is stored and described in *attributes*. An attribute consists of key-value pair, while *value* can be one of *String, Date, Integer, Float,*

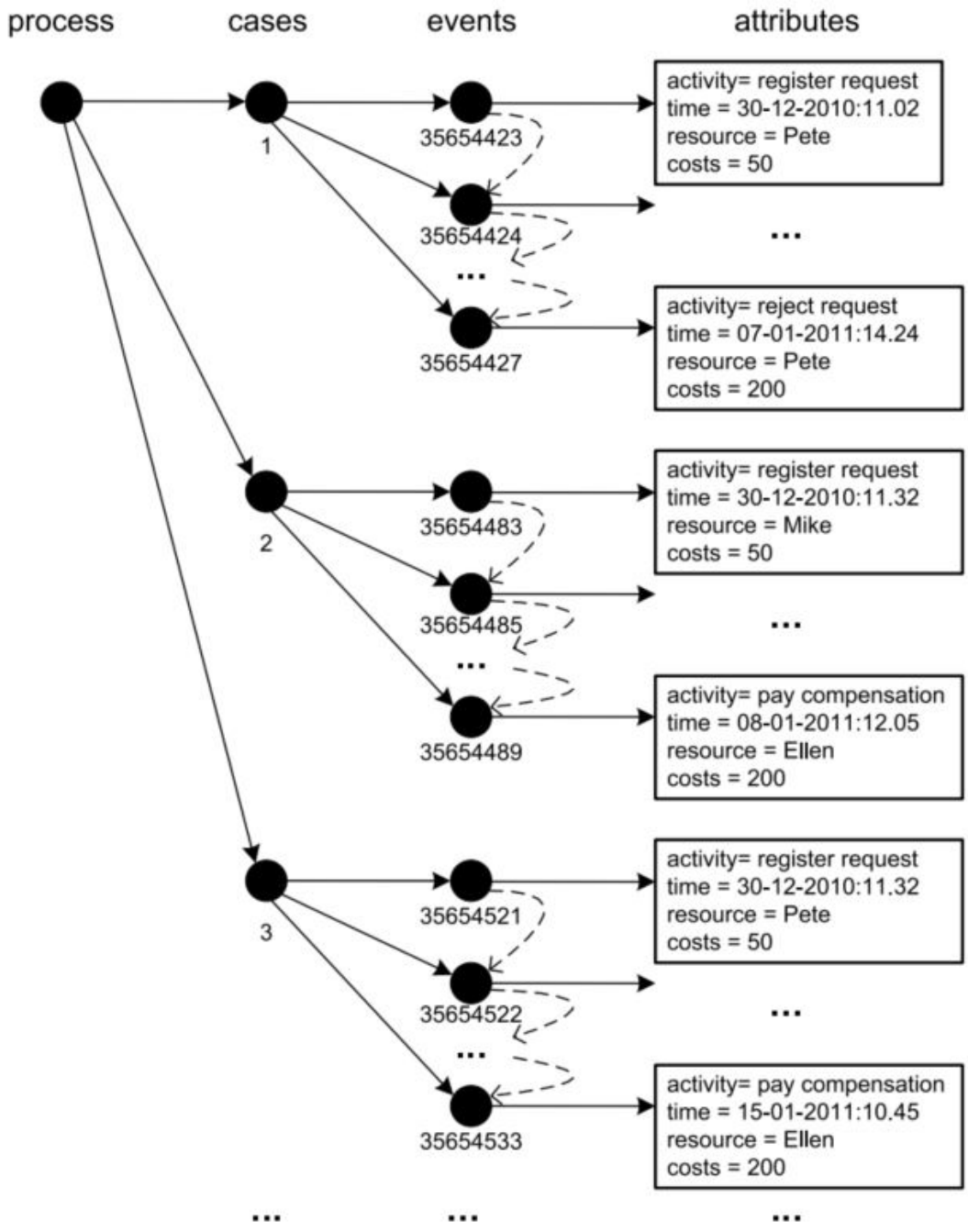


Figure 2.2: Structure event log [36]

Boolean, Id, List or *Container* data type. With string-based *key*. The meta-model of XES can be seen at Fig.2.3. As can be seen, the metamodel (Fig.2.3) is the formalised standard of event log that Van der Aalst [36] defined previously.

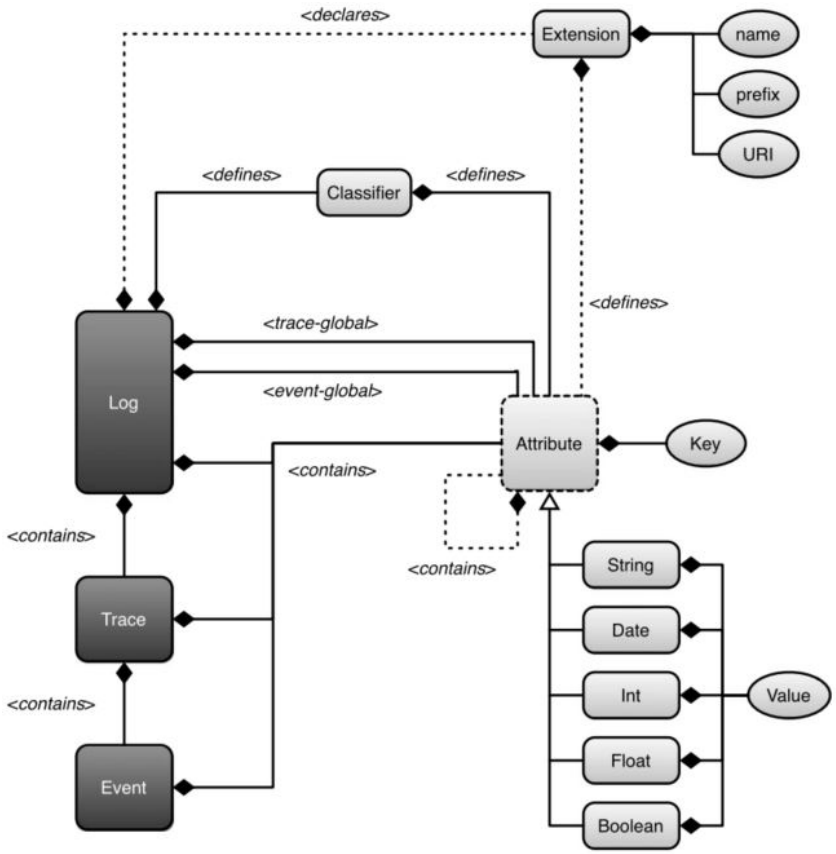


Figure 2.3: Meta-model of the XES standard [20]

2.2.1.3 Event log type

In [36] data in event log can be classified into "*pre mortem*" and "*post mortem*" data. *Post mortem* data refers to information of cases that have completed or historical data. The objective of this data not to influence the current process, but to improve the process and auditing. Meanwhile, *pre mortem* data refers to data that is still running or "alive" (pre mortem). It is possible to exploit these type of data to ensure correctness or improve the effectiveness of the processes.

Post mortem data is suitable for offline process mining, for example discovering the control-flow (Section.2.2.3) based on historical process data. However, for online process mining it is necessary to use a combination of "*pre mortem*" (current) and "*post mortem*" (historical) data. We can produce a predictive model based on historical data and improve running case using the predictive model, for example, to predict the estimation time needed for running the processes. Based on the data type, process mining itself refined into two types of models, "*de jure models*" and "*de facto models*". *De jure models* refer to the normative model that accepted or perceived by stakeholders of how things should be done or handled. While *de facto models* refer to the descriptive model or how it is currently done or handled based on captured reality. These type of data and models are illustrated in Fig.2.4. As can be seen at the figure, there are two arrows. De facto models are derived from reality (right downward arrow), and that de jure models aim to influence reality (left upward arrow), and in between, there are ten activities that are grouped into three categories: cartography, auditing, and navigation.

Cartography refers to how the process model can be seen as the "map" (cartography)

that describes the operational processes of organisations. In order to do that abstraction or blueprint (process model) is needed. Thus, activities in these categories mainly targeted to produce a process model, enhance it or diagnose the model. *Auditing* refers to a set of activities that are used to check if business processes that are executed within certain boundaries set by managers, governments, and other stakeholders. Lastly, *navigation* refers to activities that needed to navigate running and future business process, for example, exploration of business process at runtime, make a prediction model of the business process, and produce recommendation based on the predictive model.

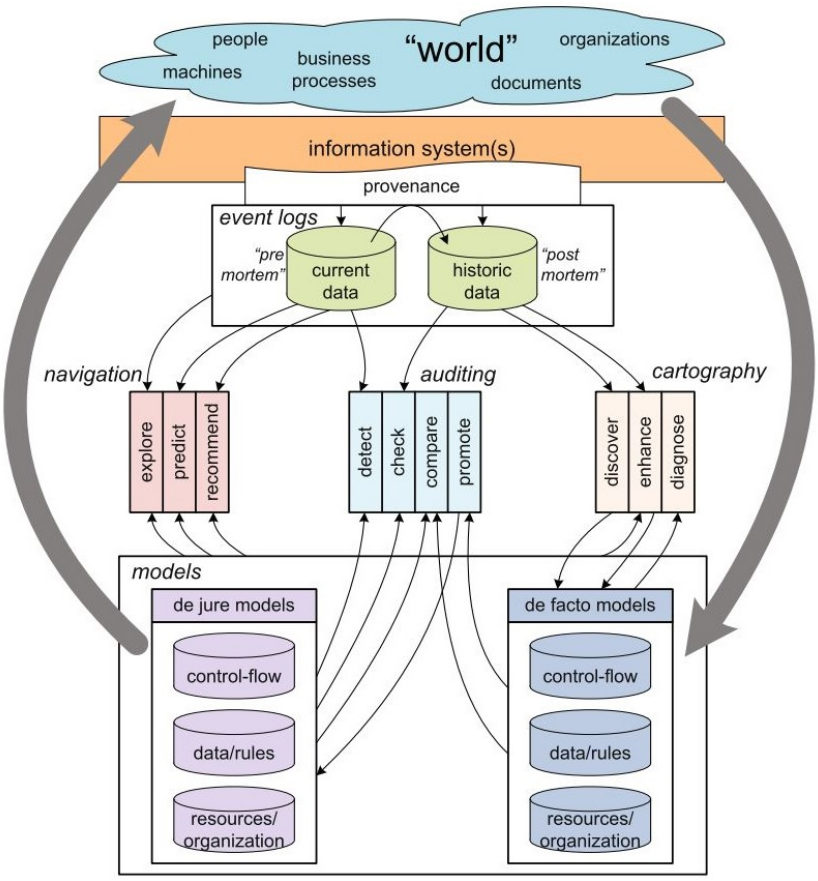


Figure 2.4: Process mining framework [36]

2.2.1.4 How to produce an event log

Van der Aalst [36] in his book also explained the process of how the logs are created. In this research we focused on "post mortem" or historical data, as can be seen at Fig.2.5 the process of event log creation needs to follow multiple steps and pre-processing until it can be used in the process mining. As described in the figure, it began with multiple data sources generated by application systems or data warehouse. The data sources are extracted using coarse-grained scoping, and they are converted into standardised event logs (in XES, MXML format). It is then filtered in more fine-grained scoping to produce filtered event logs, after that it can be used in the process mining.

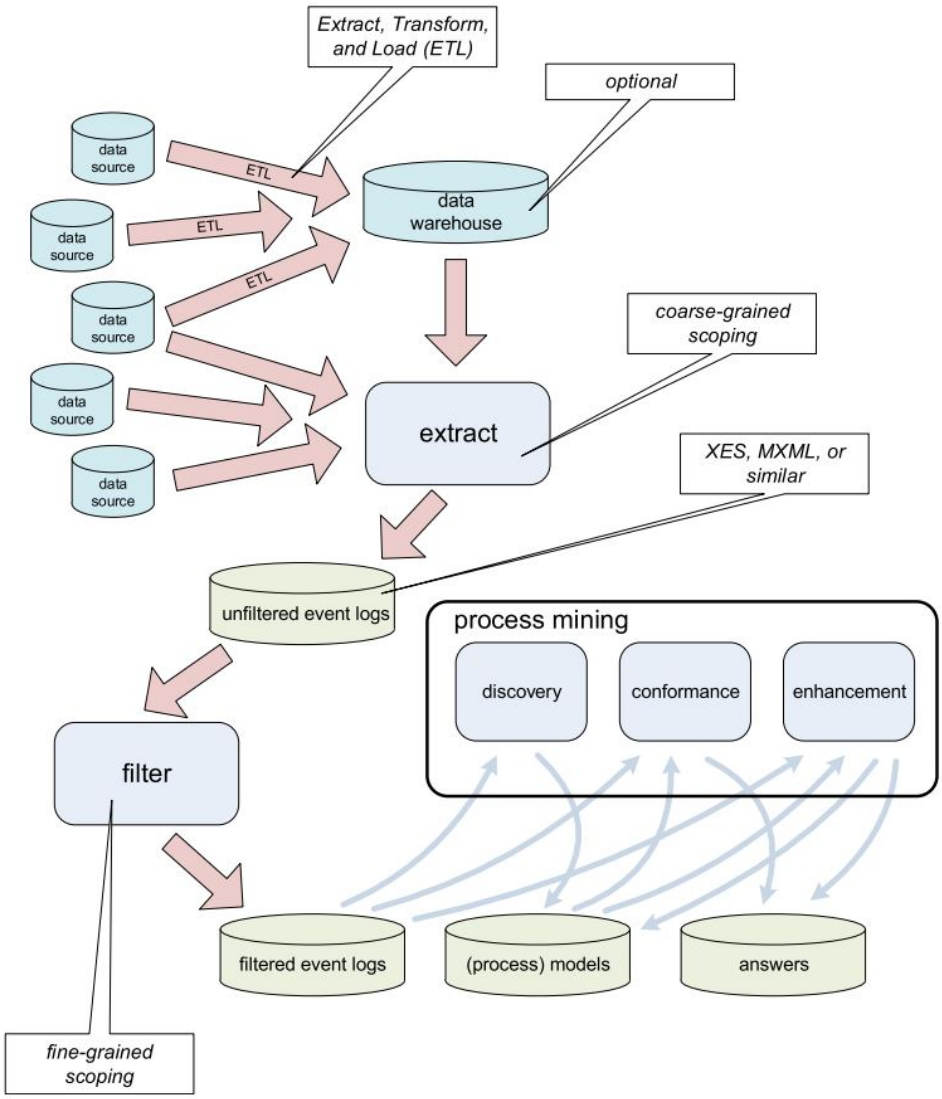


Figure 2.5: Getting process mining data from heterogeneous data source [36]

2.2.2 Is process mining algorithms can be used to produce automated EA model documentation

In this section, we found data sources available for the automation process, with its conversion mechanism. We also identify a list of fields that used to conduct the automation processes. Furthermore, we also map the fields into appropriate elements and relationships of Archimate based on literature and our conjuncture.

2.2.2.1 Data sources and automation process

RQ1b: *What are suitable data sources that can be used in automated EA model documentation?*

RQ1c: *What are the automation processes used in automated EA model documentation?*

Farwick et al. [13] in their research discovered a list of productive systems that contains relevant EA information. There are at least six information sources that might be available for Is process mining algorithms can be used to produce automated EA model documentation: *Network monitor and scanners*, a tool to gather network activities using network scanners and sensors. *Configuration management database*, a tool to collect operational data in an

organisation that complies with ITIL. *Project portfolio management tools*, a tool for managing project portfolio. *Enterprise service bus*, central mediating entities for inter-application communication. *Change management tools*, a tool to optimise implementation of changes in the IT-landscape. Lastly, *license management tools*, a tool to manage acquired software licenses.

The research from Farwick et al. [13] align with research of Holm et al. [22] and Buschle et al. [9]. Buschle et al. [9] determine Archimate informational model in business, application and infrastructure layer, the input for their research is SAP PI, Enterprise Service Bus (ESB) software that hosts on the cloud. They were able to relate SAP PI's components to Archimate's elements and relationship. They also devise transformation rules to facilitate those conversions. Although their research focused on SAP PI, it is adaptable to other ESB software. However, there are limitations in their research, as they stated in their research that SAP PI is very technological-oriented, thus limiting the input for business information related. Holm et al. [22] in their research also able to implement process mining algorithms can be used to produce automated EA model documentation using a network scanner. They defined elements and relationship of Archimate that relatable to data that gathered using network scanner and propose a tool to map between Archimate constructs and data source from the scanner. Both research from Buschle et al. [9] and Holm et al. [22] proved the options to used Network monitor and scanners and ESB for data source input in generating automated EA model.

In addition to Farwick et al. [13] research, we also discover other data source available to creating Is process mining algorithms can be used to produce automated EA model documentation. Van Langerak et al. [42] in their research implement mechanism from process mining to develop Business Process Cooperation Viewpoint in Archimate. They utilised Process-Aware Information System (PAIS) to record reality in the form of audit trails or event logs and using that logs as input to generate Archimate constructs. However, they did not provide tools for converting the log into an EA model, and it is unclear how they correlate between process models and EA models.

2.2.2.2 Relevant fields and mapping

RQ1d: *What are the relevant fields and the mappings between fields and Archimate constructs?*

There is a list of elements and relationship that can be possibly converted into EA models using a PAIS' log [42], Enterprise Service Bus [9] and network scanner [22]. In this research, we limit our research to only use Archimate as our EA framework. The list is not exhaustive since we based the list on the literature that we searched, and the list can be found at Appendix. A.

2.2.2.3 Business Layer

In the business layer, based on the literature, we could convert data sources into *Business Actor, Business Collaboration, Business Process*, and *Business Service*. Van Langerak et al. [42] in their research argued that *Organisation* and *Department* are specialisation of *Business Actor*, an *Organisation* has a hierarchical structure of *Departments*, and delivers some *Business Service*. A *Business Service* is implemented by one or more *Business Processes*. *Activities* in a *Business Process* form *Cooperation* (or *Business Collaboration*). A *Cooperation* is always initiated and concluded by an *Activity*. Using an event log, *Trace* can be used to produce *Business Service* which implements *Business Process*. While *Event* are raised by executing *Activities* and can be used to reconstruct *Cooperation*. Also *Resources* as one who executes *Activities* can be used to create *Business Actor* through *Department* and *Organisation*. The conceptual model of their idea can be seen in Fig.2.6.

While [9] stated that elements of *Business Layer* are quite difficult to reconstruct, as we quote from their research "*although SAP PI's element are meant to implement business functionality a reconstruction of business information is commonly hindered by the strong technology focus*". Some business elements that can be used to describe commutative goals such as (meaning and value) is absent from SAP PI's. In addition, elements such as *Business*

Object are not directly included in SAP PI. Other elements, such as *Representation* can be reconstructed with implicit information, as in SAP's, it derives from technologies data such as e-mail. Overall, in the research, some element can be implicitly reconstructed while others are not possible because of the absence of the data. Archimate elements that can be covered by SAP PI can be seen at Figure 2.7. In [22] it is possible to reconstruct a *Business Actor* from the data, as they described that a scanner collects all user accounts of a computer system, however, if one wants to relate these actors to different actor, i.e. department, it requires additional effort from the modeller to perform the translation.

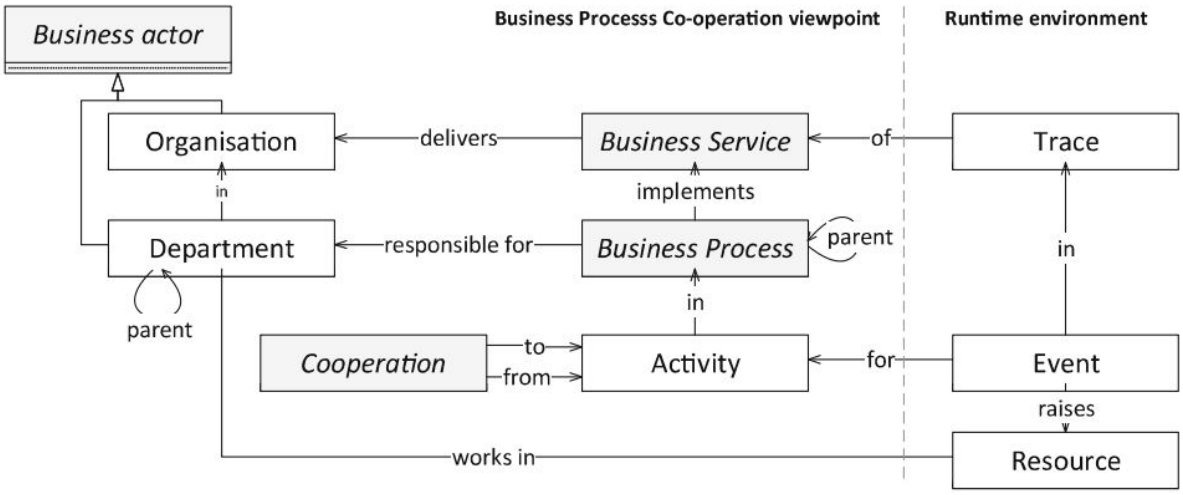


Figure 2.6: Conceptual model of runtime business architecture [42]

2.2.2.4 Application Layer

In the *Application Layer*, we can reconstruct *Application Component*, *Application Collaboration*, *Application Interface*, and *Data Object*. Buschle et al. [9] stated that SAP PI provides *software components* and *software products* to fulfil the information needed for *Application Component*. Also *Application Collaboration* is reconstructed by temporary configuration of two or more *Application Components*. *Application Interface* is broadly similar to SAP PI's *enterprise service interface*. In SAP PI there are no elements available to describe behaviour elements, such as *Application Service*. However, they argued that it can be derived indirectly using *interface* and description of *operations*. Lastly, *Data Objects* is similar to SAP PI's *data types*. Their conjuncture regarding elements of SAP PI that can be converted into EA model can also be seen in Fig.2.7. In [22], *Application Component* can be reconstructed using data of various application components that gathered by the scanner, for example, different ERP system modules, application clients such as Adobe Reader. If an application is running on an end-point (i.e., a port), it can provide information for *Application Interface*. In regards to [9], they were unable to reconstruct it since the SAP PI does not provide data to do it. It implies that it actually can be reconstructed as long as the data available. The same reasoning also applicable in [22], there are no data available in the scanner to record behaviour processes.

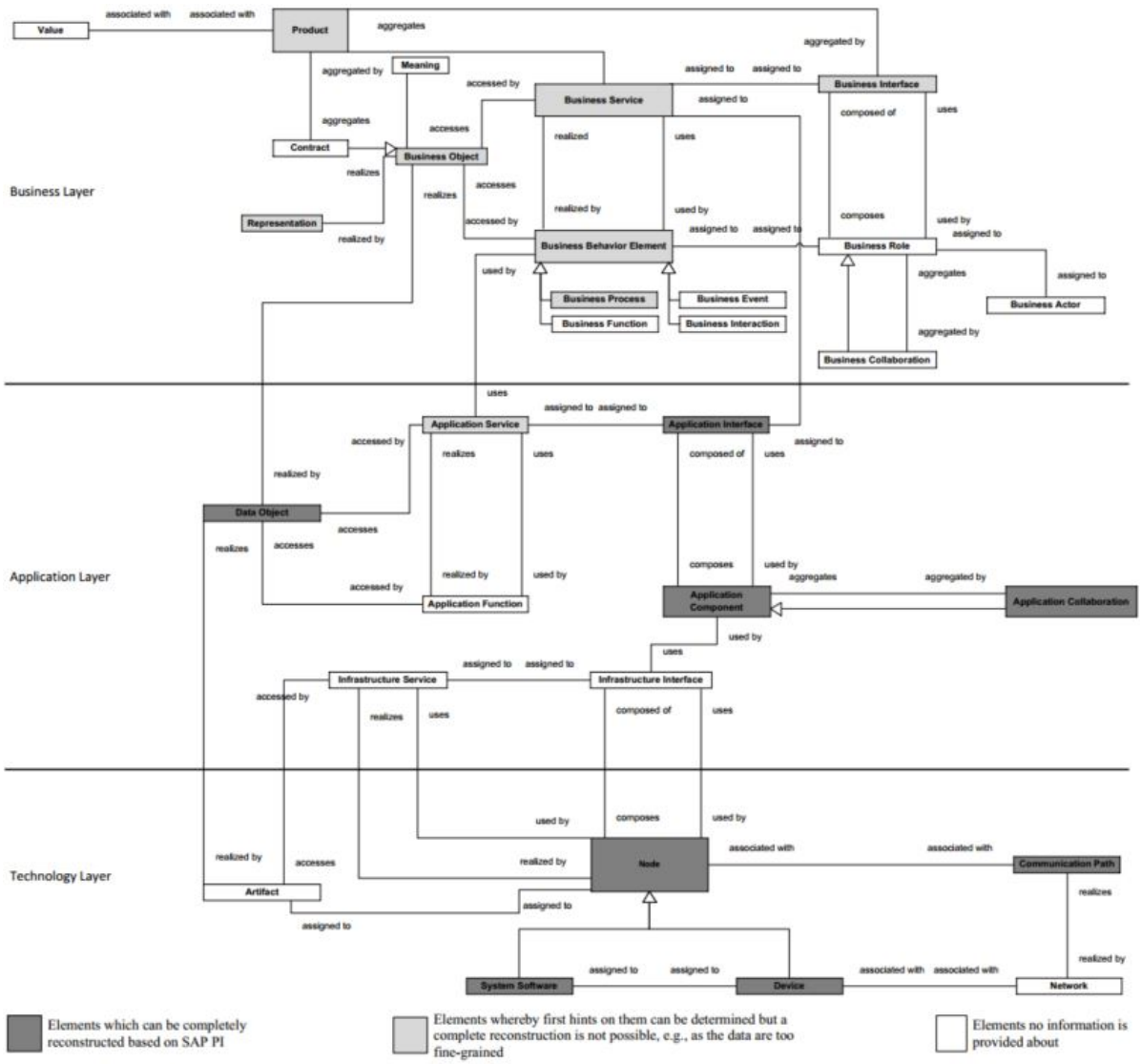


Figure 2.7: Archimate information model elements covered by SAP PI [9]

2.2.2.5 Technology Layer

In this layer, we can reconstruct *Node, Device, System Software, Technology Interface*, and *Communication Network*. Buschle et al. [9] stated that SAP PI's *computer system* can fill in for a *Node*. A subset of installed system software registered at System Landscape Directory of SAP PI can be used to represent *System Software*. While *Communication Network* can be reconstruct using an underlying physical medium of each service invocation by SAP PI. However they did not provide information on how a device can be recreate using SAP PI, yet they put the information in the picture (Fig.2.7). In [22] a *System Software* is represented by the identification of several types of system software, for example, web servers and operating systems. *Infrastructure Interface* can be fill in using the protocol (i.e., SMTP) and port (i.e., 8080). While *Device* is represented by hardware or IP address of a system. Holm et al. [22] also argued that IP address could be used to represent a *Network*.

2.2.2.6 Relationship

Based on the literature we can define relationship such as *Composition, Aggregation, Assignment, Realization, Serving, Access, Triggering* and *Association*. Buschle et al. [9] in their research suggest that *Composition* and *Access* was used to defined relationship between *Ap-*

plication Interface and *Application Component*, while *Aggregation* used to relate *Application Component* and *Application Collaboration*. While Assignment was used in relationship between *System Software* and *Device*. *Node* can be related to *Communication Path* through *Association*. In addition, [42] implied that they used *Composition, Realization, Serving,* and *Triggering* in their research. While [22] stated that they used Assignment to define relationship between Device and System Software, and *Used by* (*Serving*) to define relationship between *Application Component* and *Infrastructure Interface*.

2.2.3 Algorithms

RQ2a: *What are algorithms that are used in Process Mining to convert a log into a process model?*

2.2.3.1 Process models

2.2.3.1.1 Petri-Net and workflow-net

Definition 5 (**Petri net** [36]). *A Petri net is a triplet N = (P, T, F) where P is a finite set of places, T is a finite set of transitions such that* $P \cap T = \emptyset$*, and* $F \subseteq (P \times T) \cup (T \times P)$ *is a set of directed arcs, called the flow relation. A marked Petri net is a pair (N, M), where N = (P, T, F) is a Petri net and where* $M \in \mathbb{B}(P)$ *is a multi-set over P denoting the marking of the net. The set of all marked Petri nets is denoted* $\mathcal{N}$*. The Petri net shown Fig. 2.8 can be formalized as follows: P = {start, c1, c2, c3, c4, c5, end}, T = {a, b, c, d, e, f, g, h}, and F = {(start, a), (a, c1), (a, c2), (c1, b), (c1, c), (c2, d), (b, c3), (c, c3), (d, c4), (c3, e), (c4, e), (e, c5), (c5, f), (f, c1), (f, c2), (c5, g), (c5, h), (g, end),(h, end)}*

Petri net is a bipartite graph consisting of *places* and *transitions*, interconnected by directed arcs [36]. Petri net is one of the most basic process model that being used as an output of process mining algorithms, one of the algorithms that produced WF-net (Petri net with start and end place) is alpha miner. Petri net also depends on the token to ensure that the process model replayable / can be simulated for each cases of event log. There also split and join rules that inherent in petri net, this is useful to denotes if some events have concurrent activities or some execution path is exclusive for some events.

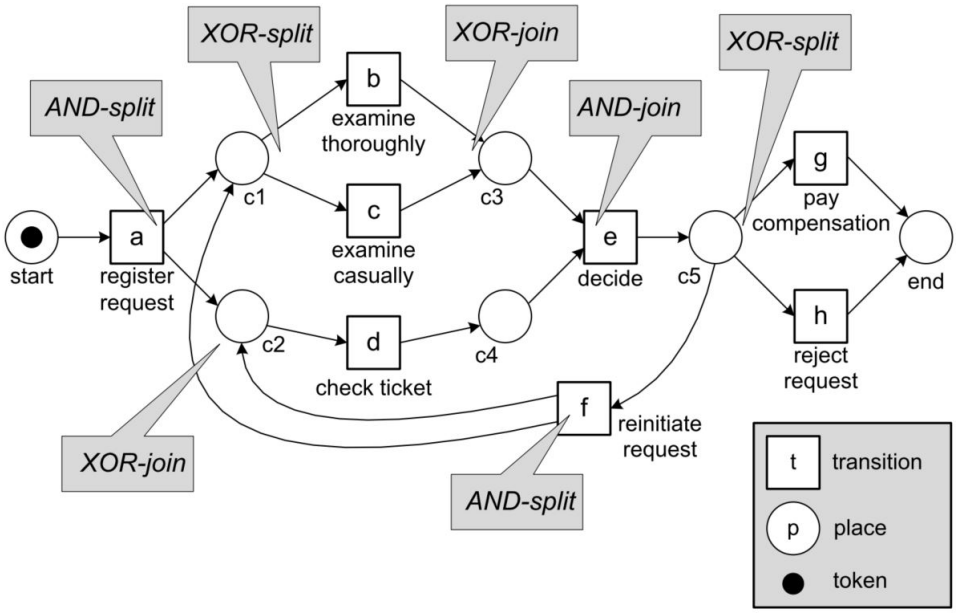


Figure 2.8: Workflow net [36, p.37]

Definition 6 (**Workflow net** [36]:). *Let N = (P, T, F, A, l) be a (labeled) Petri net and $\bar{t}$ a fresh identifier not in $P \cup T$. N is a workflow net (WF-net) if and only if (a) P contains an input place i (also called source place) such that $\bullet i = \emptyset$, (b) P contains an output place o (also called sink place) such that $o\bullet = \emptyset$, and (c) $\bar{N}$ = $(P, T \cup \{\bar{t}\}, F \cup \{(o, \bar{t}), (\bar{t}, i)\}, A \cup \{\tau\}, l \cup \{(\bar{t}, \tau)\})$ is strongly connected, i.e., there is a directed path between any pair of nodes in $\bar{N}$.*

A workflow net (WF-Net) is a Petri net with a single start place and a single end place that represent the start and end state of a process.

2.2.3.1.2 Dependency graph

Heuristic miner produces dependency graph (Definition.15) as its process model [44], the dependency graph represents all the dependencies found in the log. Fig.4.6 is an example of dependency graph that resulted from a heuristic miner processing. As can be seen at the figure, it has rectangle to represent an activity with its frequency occurrences, the arc represent dependency path between activities, the arc also has label that represent dependency factor and its frequency. The result of *business process discovery* algorithm is similar to dependency graph, and instead of rectangle as an activity, we substitute it with *business process* element, and we convert arcs to *triggering* relationship. Example of the converted graph can be seen at Fig.4.6.

Definition 7 (Dependency Graph [4]). *Given a set of activities V and a log of executions L of the same process, a directed graph G_{VL} is a dependency graph if there exists a path from activity u to activity v in G_{VL} if and only if v depends on u.*

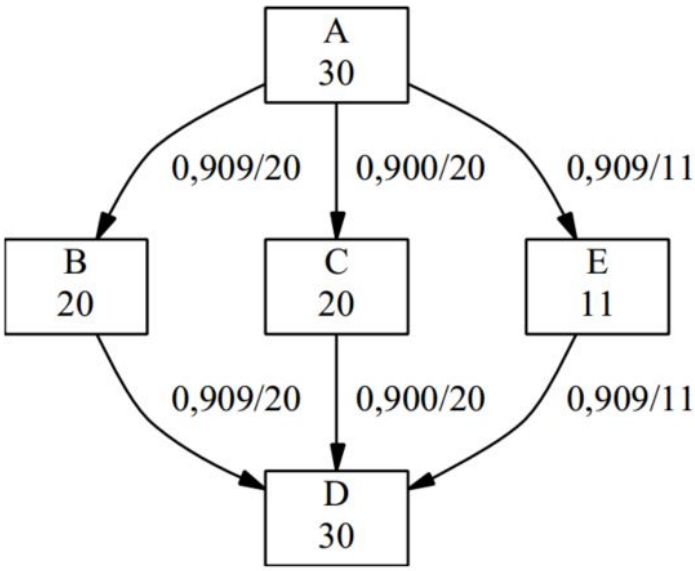


Figure 2.9: Example dependency graph [44]

2.2.3.2 Algorithms

There are three types of Process Mining: *discovery* [6–8, 26, 29, 30, 32, 36, 38, 40, 43], *conformance* [8, 26, 32, 36, 38], and *enhancement* [8, 32, 36]. *Discovery* is a technique to process an Event log without using apriori information, while *conformance* is a mechanism to check if the reality as written in the log conforms to an existing process model and vice versa. Lastly, enhancement, a mechanism to extend or improve an existing process model using the actual process information that recorded in a log. In the process discovery itself there are several perspectives to emphasise which behaviour that the techniques want to

observe, such as *control flow* [8, 26, 29, 30, 32, 36, 40], *organisational* [7, 8, 26, 29, 30, 32, 36, 40], *performance / time* [8, 29, 30, 32, 36], and *data / case* [26, 36, 40]. The control flow perspective focuses on the ordering of activities, to find a good characterisation of all possible paths of processes. The organisational perspective discovers the hidden information regarding resources in a log, which actors (i.e. people, system, roles, and departments) are involved and relationship between them. The organisational perspective also has an objective to structure the organisation by classifying people in regards of roles, organisational unit, or social network. The case perspective focuses on the properties of cases. A case can describes a process or originator that working on it, or it can also describe values of corresponding data elements, such as number of products ordered. The time perspective explains the timing and frequency of events, discovering bottleneck, measure service levels, monitor utilisation of resources, and predict the remaining processing time of running cases.

In this study, we found out algorithms that used for each perspective in the process discovery (Appendix.B). Also, in the following sub-sections we described in more detail algorithms that we think can be potential algorithms for converting a log into an EA model.

2.2.3.3 Control Flow Perspective Algorithms

Control flow algorithms help to discover the sequence of processes that reside in an event log. The algorithms also able to describe the dependency of processes, correlations between those processes and the frequency of those processes occurs. Related to our study, this information will help us to uncover certain elements of the EA framework, for example in Archimate elements such as *business processes*. The algorithms also able to cluster coherent activities and limiting the processes using the high degree of correlation or frequencies that occurs, this capability will help reducing the number of elements that can be generated in an EA model, thus relieving over-complexity of the EA model.

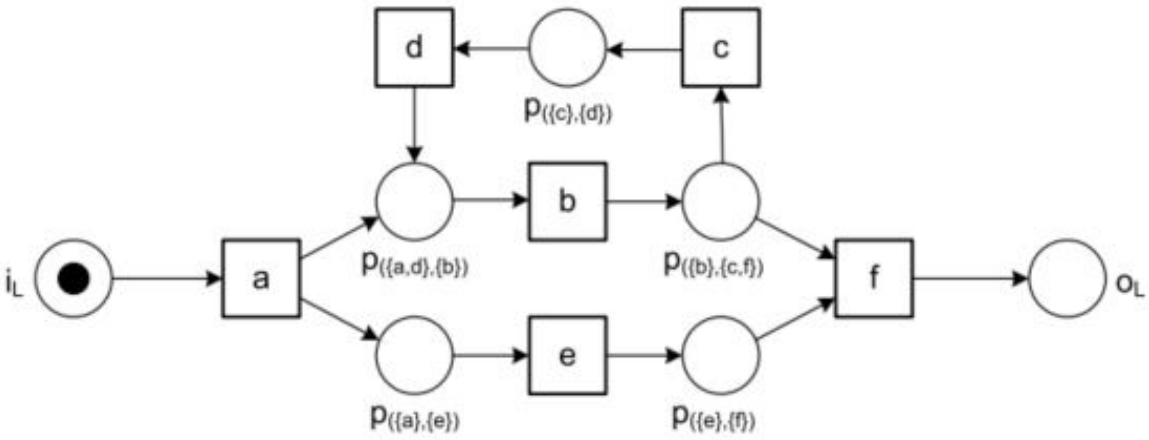


Figure 2.10: Alpha Miner result Example [36]

2.2.3.3.1 Alpha Miner

The alpha miner is one of the first algorithms that can be used to discover concurrency [36], it discovers dependency pattern between activities and describes process behaviour within the log. The alpha miner scans the event log for particular pattern [36]. It able to discern between dependency, non-dependent, concurrency relationship between events. They also enabled process patterns, to distinct if relationship between events are sequence, XOR/AND-split or XOR/AND-join pattern. Then algorithm also ensure that the process model is re-playable or enable to simulate when its needed for analysis. The algorithm produces a workflow net: a bipartite graph consisting of *places* and *transitions* interconnected by directed arcs and has a single start and end place. In the Fig.3.3 it can be seen a sample of behavioural activities based on a given log, for example, the whole process begin with activity *a* and end in *f*, with rectangles as transitions, and circles as places and connected with directed arc. The circles

also represent process pattern, for example from activity a to d is connected with dependency relationship in sequence, while b and e connected to f with XOR-join relationship. The algorithms is quite simple and easily to implement, however there are certain limitation for this simplicity, it has difficulty to process infrequent or rare behaviour (noise), logs that only contain a little or few events (incompleteness) or log with complex routing constructs.

2.2.3.3.2 Heuristic Miner

Heuristic Miner is an algorithm that able to deal with noise and exceptions. It also have capabilities of alpha miner to discern dependency, non-dependent, concurrency relationship between events. It also focus on frequencies of events and sequences, and it enables users to concentrate on the main process flow instead of on every detail of the behaviour that appears in the process log. Heuristic Miner produces a Causal Net, a graph that represents activities as nodes and dependencies as arcs and also dependency graph, directed graph that represent dependent relationship between activities. For example, in Fig.2.11 sample of a behavioural analysis using Heuristic Miner. It can be seen the algorithm able to discover frequencies of processes. In the activity of "*ArtificialStartTask*", it has 352 occurrences, and 36 of those occurrences happens to be followed by "*Yearly checkup OC Obst/Gyn*". The algorithm also able to extract correlation between processes, as shown the dependency factor between "*ArtificialStartTask*" and "*Yearly checkup OC Obst/Gyn*" is 0.973 or 97.3% "*ArtificialStartTask*" will be directly followed by "*Yearly checkup OC Obst/Gyn*".

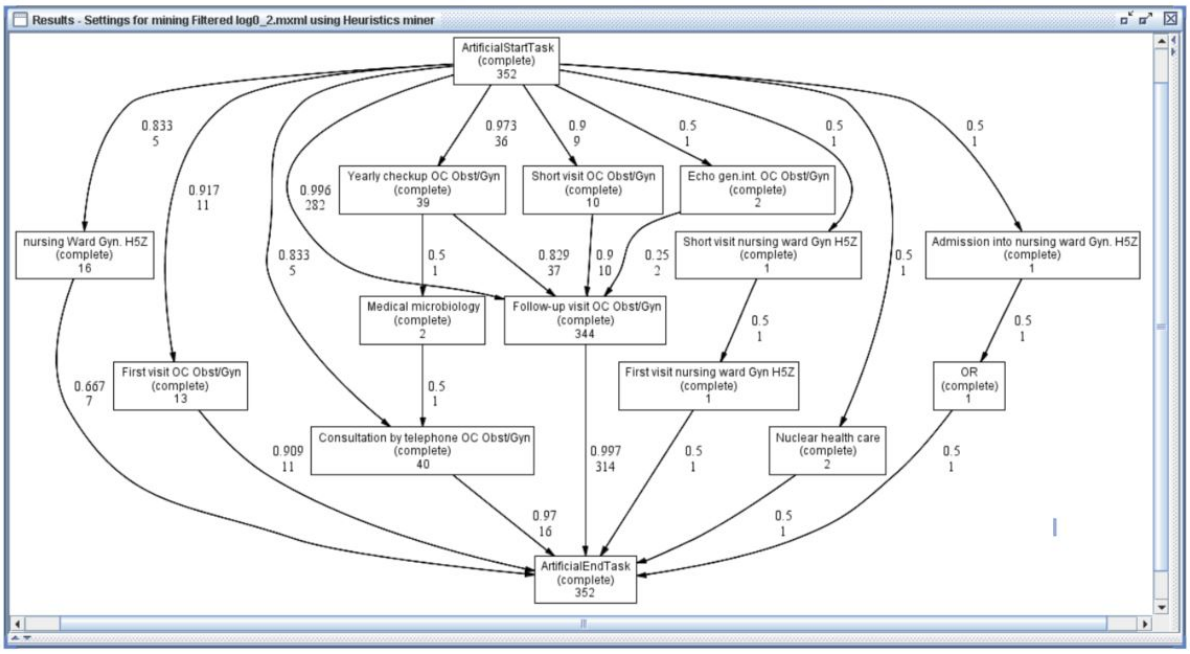


Figure 2.11: Heuristic Miner Example [30]

2.2.3.3.3 Fuzzy Miner

Fuzzy Miner able to generate process models from the huge number of activities and highly unstructured behaviour. The Fuzzy Miner combines abstraction and clustering techniques to produce a high-level view and emphasising the most important details. In the Fig.2.12 it can be seen the example of using this algorithm to extract behavioural process of a log. The algorithm able to discover frequency of processes, it can be seen at the thickness of the arc, for example, "*1*" to "*61*" is more frequent than "*495*" to "*61*". The algorithm also able to cluster coherent and highly correlated activities, it is depicted as a green hexagon.

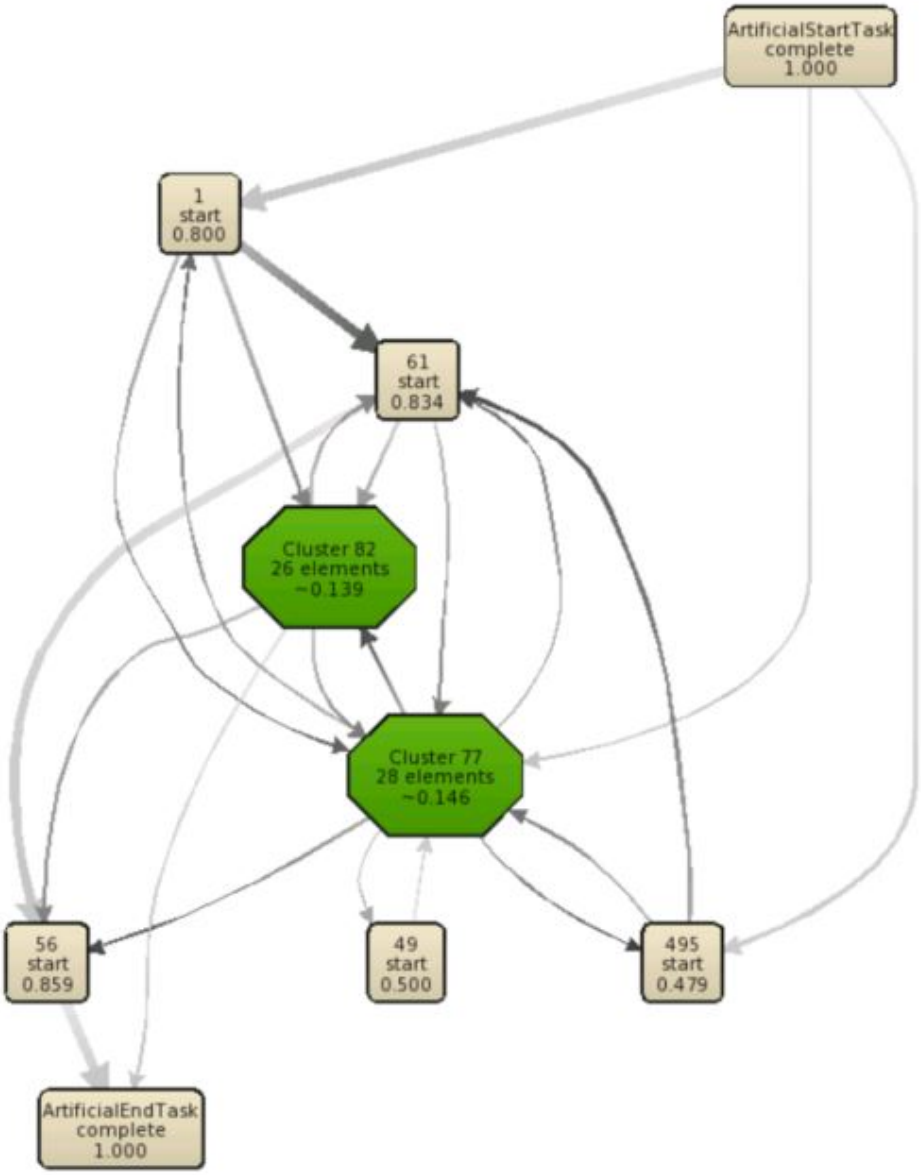


Figure 2.12: Fuzzy Miner Example [8]

2.2.3.3.4 Genetic Miner

Alpha Algorithm, Heuristic, and Fuzzy Mining able to discover process models in a direct and deterministic manner, it is different with Genetic Miner, it provides process models in evolutionary approach, using a technique from the field of computational intelligence. As can be seen at Fig.2.13 there are four main steps: (a) initialisation: creating an initial population (b) selection: determine the quality of an individual process model (determine the fitness level) and select the best individual to the next generation (c) reproduction: selecting a parent individuals and used to create new offspring (process model) and modified the resulting children with mutation (i.e., randomly adding or deleting a causal dependency) (d) termination: terminating evolutionary process when a suitable process model is found. The algorithm can also deal with noise and incompleteness. However, the algorithm is not very efficient for larger models and logs as the algorithm requires a very long time to discover an acceptable model [36].

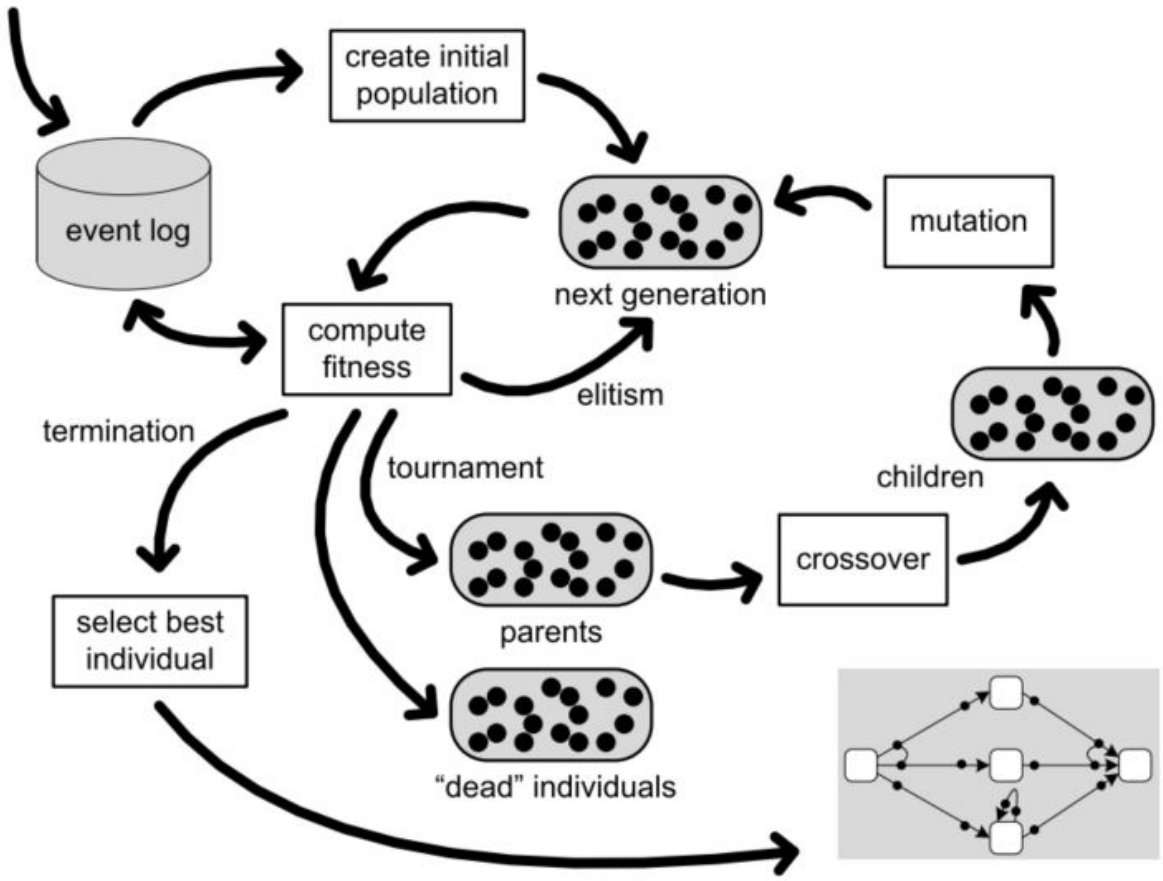


Figure 2.13: Genetic Process Mining Overview [36]

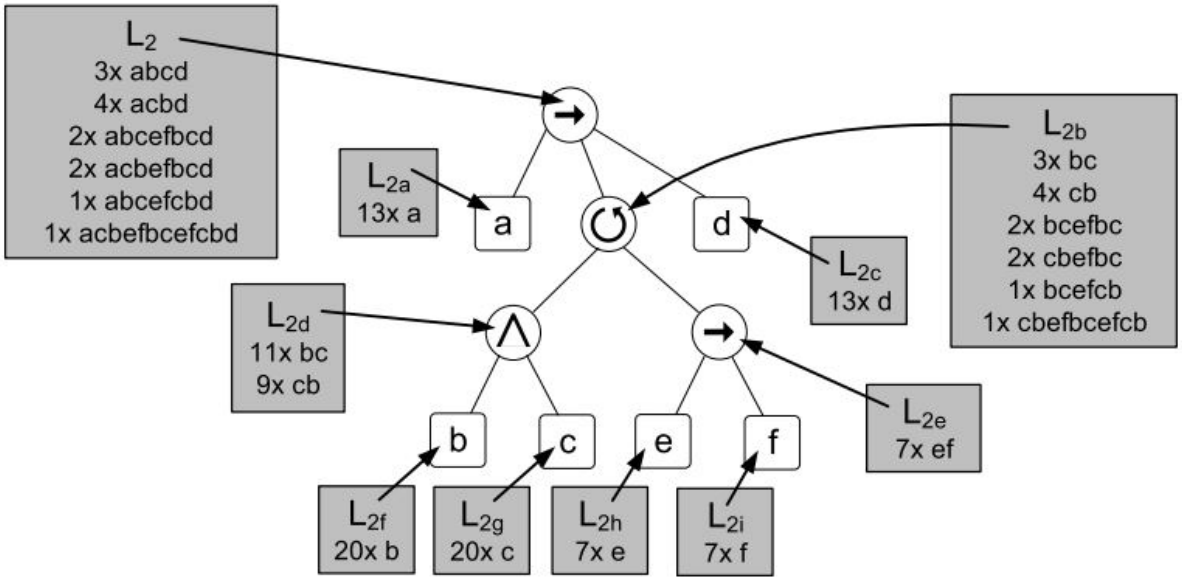


Figure 2.14: Inductive Miner Sample [36]

2.2.3.3.5 Inductive Miner

Inductive miner is currently one of the leading process discovery algorithm [36]. The algo-

rithm is highly extensible, it also able to handle infrequent behaviour and deal with huge model and logs, the algorithm also ensures formal correctness criteria, for example, ability to rediscover the original model. While the alpha miner produces WF-Net, Inductive miner produces equivalent process tree. The Inductive miner is processing an event log in an iterative manner, splitting the log into sub-logs and create the directly-follows graph for each sub-logs. In Fig.2.14 it can see the sample of the process tree as a process model result when processing an event log using Inductive Miner. Each node is the result of observation of sub-logs that was mentioned earlier.

2.2.3.4 Organisational Perspective Algorithms

Organisational algorithm focus on who perform what activities and express more into the relationship between originators (a person who conduct some activity). There are two algorithms that we highlight in this section, organisational and social network analysis miner. In our study, both algorithms can be used in the future research. Elements of Archimate, such as *business actors* and *business roles* can be discovered using these algorithms.

2.2.3.4.1 Organisational Miner

Organisational miner is the algorithm to derive an organisational model from event logs. It can derive a group of originators that have similar characteristics in process execution and discover the relationship between mined groups and the tasks. In Fig.2.15 it can be seen the sample of using this algorithm to determine the role of each originator that lie in a given log.

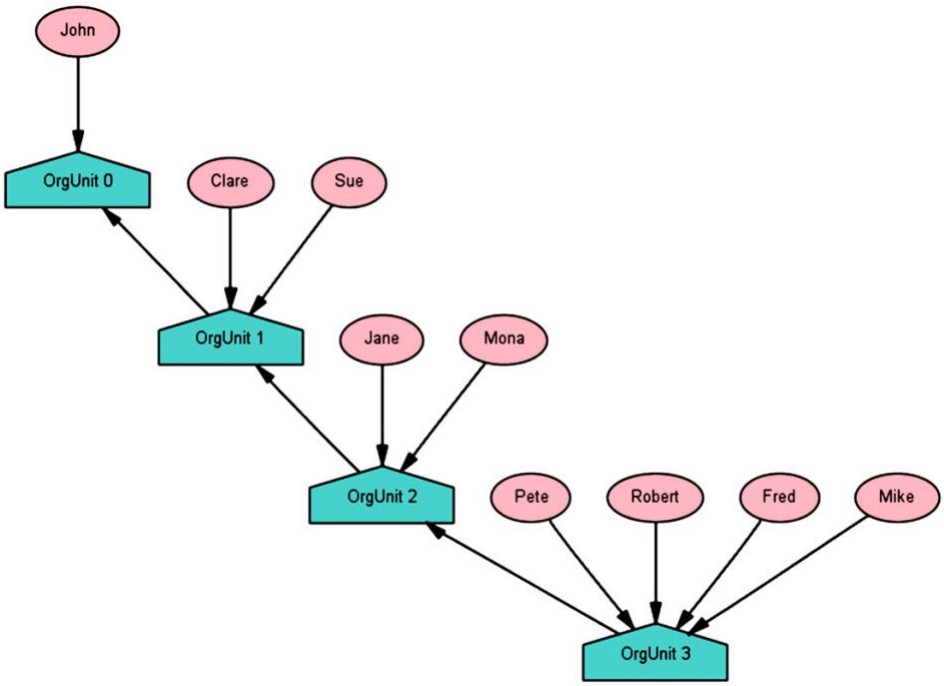


Figure 2.15: Organisational Miner Sample [33]

2.2.3.4.2 Social Network Miner

The social network analysis algorithm is the algorithm that able to derive an interpersonal relationship between originators in a given log. It can describe *handover of work*: detecting the likelihood of handover of work from one originator to other given the causality relationship of activities. In the Fig.2.16 it can be seen the handover of work from one originator to other and the density of work relationship of each originator.

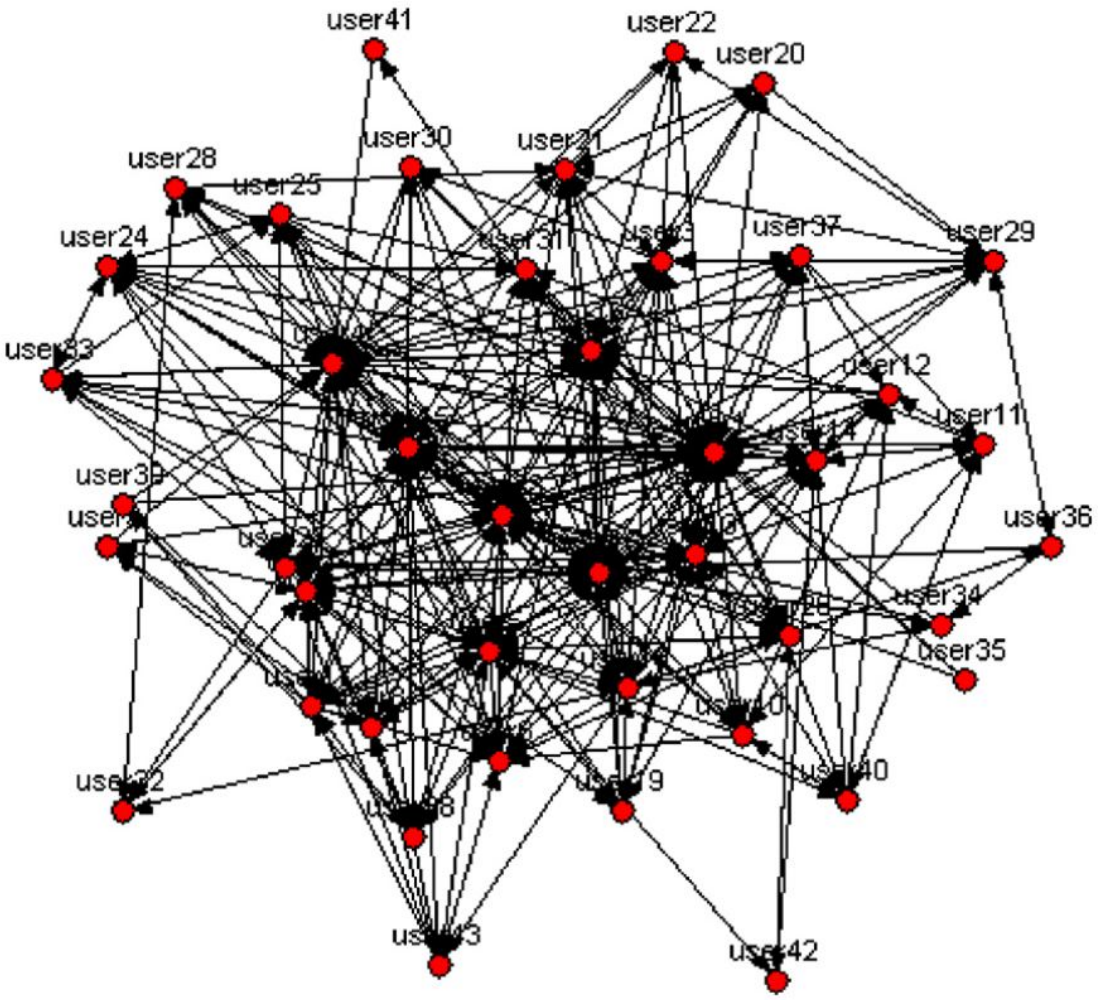


Figure 2.16: Social Network Analysis Sample [40]

2.2.3.5 Data and Performance Perspective Algorithms

Based on the literature finding, we did not find relevant algorithms that might useful for our research. We found the dotted chart, a technique to describe events and represent data attributes in two-dimensional plane. However, for this research, we seek the algorithms that enable users to extract process models from logs, while mostly algorithms of this perspectives seek attribute information that lies in logs, such as duration of activities, resources that execute the activities. Overall, we think it is irrelevant to use this perspectives' algorithm for our research unless we want to describe the performance of activities in our EA model.

2.2.4 Conversion pattern

RQ2c: *What are Archimate elements that can be used to represent process models?*

In process mining, there are various perspective to observe the behaviour of processes in an event log. In this research, we picked control-flow perspective as the main perspective for automated EA model documentation. Control-flow perspective is able to discover a sequence of processes in an event log. We chose control-flow since we want to discover the workflow of processes of various systems. We can identify the flow of processes from the beginning to the end. For example, in e-commerce case, we can identify an order number of certain products. Hence, using order number, we can trace the status of the order from the very beginning (order placement), to various step of systems (payment, delivery) to the end (bookkeeping).

Hence, using control-flow we can achieve this objective.

We perceive each step in the e-commerce case is an event, and for an event, we can attach various information as attributes, such as the name of the system, a user who execute the events, time its been executed, etc. Thus, identifying the event is the most crucial part in automated EA documentation using process mining. As we discussed before there are multiple algorithms that we can choose, however for this research we chose the simplest one. In the beginning, we chose alpha miner as the algorithm provide a basic algorithm to control-flow perspective, but we also realised that alpha miner has limitations in recognising infrequent/rare, incomplete, or complex routing case. Thus, we chose heuristic miner for this reason. Heuristic miner also simpler than other algorithms that more mature and advance, so we believe it is easier to implement in future works. Moreover, heuristic miner able to address the problems that raised in the alpha miner, so this is a trade-off that we pick. Heuristic miner is able to generate various type of process model, such as Causal net, dependency graph. However, in this study, we pick the dependency graph as a process model for referencing to Archimate. Again, we pick the graph to easier implementation.

The comparison process that we chose is quite simple, we identify the definition of elements of the process model, and we pick Archimate elements that closely represent that definition. Based on the definition of the dependency graph (Definition.15), the graph consisted of a set of activities and connected to the arc that represents the dependency between activities. The *activity* itself is referred to "a well-defined step in the process." [37]. And we interpret it as *Business Process*, "A sequence of business behaviors that achieves a specific outcome such as a defined set of products or business services." [18]. In our opinion activity is a step of the process, the process in this context is a business-related process, and it closely fit business process that defines a sequence of business behaviour, since the process discovery also targeted to observe certain behaviour of processes in the log. Hence, we think the business process can be used to represent activity. In addition, there are multiple relationship that represent relationship between business processes, such as *specialization, composition, aggregation, serving, triggering, flow, and association* [18, p.125] and since arc in the dependency graph represent dependency relationship, thus we chose *serving* to connect between *business processes*. The *serving* relationship models that an element provides its functionality to another element [18]. Serving is one of the dependency relationship that defines in Group [18].

3

Theoretical Background

In this chapter we will discuss additional theoretical background that were used in the research that had not been explained before in previous chapters. We discuss the definition of EA, Archimate as an EA framework, popular process mining algorithms in more detail, and tools that were commonly used in process mining.

3.1. Enterprise Architecture

3.1.1 Enterprise Architecture

Enterprise architecture is a coherent whole of principles, methods, and models that are used in the design and realisation of an enterprise's organisational structure, business processes, information systems, and infrastructure [28]. While Gartner [16] defined enterprise architecture is a discipline for proactively and holistically leading enterprise responses to disruptive forces by identifying and analysing the execution of change toward desired business vision and outcomes. EA delivers value by presenting business and IT leaders with signature-ready recommendations for adjusting policies and projects to achieve target business outcomes that capitalise on relevant business disruptions. And [19] defined enterprise architecture with two separate definitions. Enterprise is the fundamental organisation of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. And architecture are: (1) A formal description of a system or a detailed plan of the system at component level to guide its implementation. (2) The structure of components, their inter-relationships, and the principles and guidelines governing their design and evolution over time.

3.1.2 Archimate

The Archimate Enterprise Architecture Modelling language is a technical standard from The Open Group. It provides a diagram standardisation that used to describe Enterprise Architectures. It presents a selection of viewpoints to address a variety of stakeholders, an integrated architectural approach that used to describe and visualise multiple architectural domains and their underlying relations and dependencies [18].

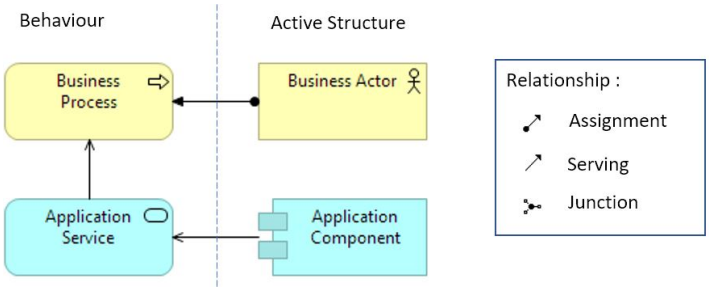


Figure 3.1: Simplified EA metamodel

In this research, we use a simplified version of the Archimate modelling language. As can be seen at Fig.3.1, we used business and application layer. With four elements, *business process*, *business actor*, *application service*, and *application component*. With addition of two relationship, *assignment* and *serving*. Using definition from [18], a business process represents a sequence of business behaviours that achieve a specific outcome such as a defined set of products or business services. A business actor is a business entity that is capable of performing behaviour. An application service represents an explicitly defined exposed application behaviour. While an application component represents an encapsulation of application functionality aligned to implementation structure, which is modular and replaceable, it encapsulates its behaviour and data, exposes services, and makes them available through interfaces. Moreover, the assignment relationship expresses the allocation of responsibility, the performance of behaviour, or execution. Next, the serving relationship models that an element provides its functionality to another element. Lastly, we also include *junction*, a junction is used to explicitly express that several elements together participate in the relationship (and junction) or that one of the elements participates in the relationship (or junction). A junction should either have one incoming and more than one outgoing relationships, or more than one incoming and one outgoing.

3.2. Process Mining

Definition 8 (Process mining [37]). *Techniques, tools, and methods to discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs commonly available in today's (information) systems.*

There are three types of Process Mining: *discovery* [6–8, 26, 29, 30, 32, 36, 38, 40, 43], *conformance* [8, 26, 32, 36, 38], and *enhancement* [8, 32, 36]. *Discovery* is a technique to process an Event log without using apriori information, while *conformance* is a mechanism to check if the reality as written in the log conforms to an existing process model and vice versa. Lastly, enhancement, a mechanism to extend or improve an existing process model using the actual process information that recorded in a log, illustration of process mining overview can be seen at Figure.3.2.

In the process discovery itself there are several perspectives to emphasise which behaviour that the techniques want to observe, such as *control flow* [8, 26, 29, 30, 32, 36, 40], *organisational* [7, 8, 26, 29, 30, 32, 36, 40], *performance / time* [8, 29, 30, 32, 36], and *data / case* [26, 36, 40]. The control flow perspective focuses on the ordering of activities, to find a good characterisation of all possible paths of processes. The organisational perspective discovers the hidden information regarding resources in a log, which actors (i.e. people, system, roles, and departments) are involved and a relationship between them, with objectives to structure the organisation by classifying people in regards of roles, organisational unit, or social network. The case perspective focuses on the properties of cases. A case can describe a process or originator that working on it, or it can also describe values of corresponding data elements, such as supplier or number of products ordered in a case of order replenishment. The time perspective explains the timing and frequency of events, discovering bottleneck, measure service levels, monitor utilisation of resources, and predict the remaining processing time of

running cases.

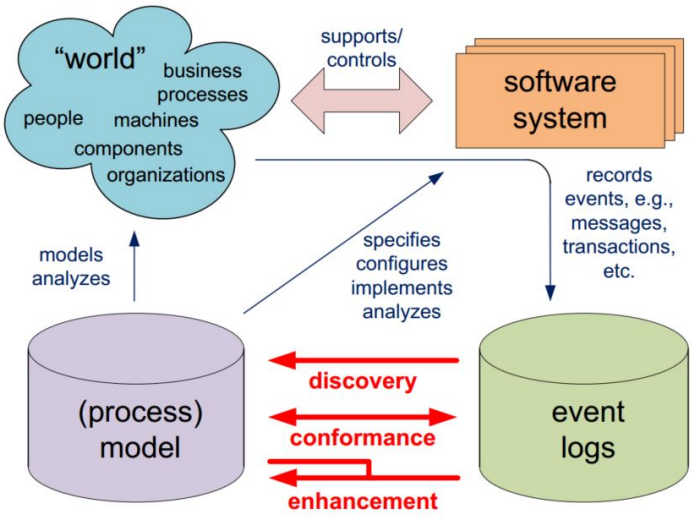


Figure 3.2: Process mining overview

3.2.1 Alpha miner

Alpha miner is an algorithm in Process Mining that tries to discover a process model from an event log and aimed at reconstructing causality of events between observed tasks.

Definition 9 (Alpha Miner [36]). *Let L be an event log over $T \subseteq \mathscr{A}$. $\alpha(L)$ is:*

1. $T_L = \{t \in T \mid \exists_{\sigma \in L}\, t \in \sigma\}$,
2. $T_I = \{t \in T \mid \exists_{\sigma \in L}\, t = first(\sigma)\}$,
3. $T_O = \{t \in T \mid \exists_{\sigma \in L}\, t = last(\sigma)\}$,
4. $X_L = \{(A, B) \mid A \subseteq T_L \wedge A \neq \emptyset \wedge B \subseteq T_L \wedge B \neq \emptyset \wedge \forall_{a \in A} \forall_{b \in B}\, a \rightarrow_L b$
   $\wedge \forall_{a_1, a_2 \in A} a_1 \#_L a_2 \wedge \forall_{b_1, b_2 \in B} b_1 \#_L b_2\}$,
5. $Y_L = \{(A, B) \in X_L \mid \forall_{(A', B') \in X_L} A \subseteq A' \wedge B \subseteq B' \Longrightarrow (A, B) =$
   $(A', B')\}$,
6. $P_L = \{p_{(A,B)} \mid (A, B) \in Y_L\} \cup \{i_L, o_L\}$,
7. $F_L = \{(a, p_{(A,B)}) \mid (A, B) \in Y_L \wedge a \in A\} \cup \{(p_{(A,B)}, b) \mid$
   $(A, B) \in Y_L \wedge b \in B\} \cup \{(i_L, t) \mid t \in T_I\} \cup \{(t, o_L) \mid t \in T_o\}$
8. $\alpha(L) = (P_L, T_L, F_L)$.

Considering event log of L_5:
$L_5 = [\ \langle a, b, e, f\rangle^2, \langle a, b, e, c, d, b, f\rangle^3, \langle a, b, c, e, d, b, f\rangle^2,$
$\langle a, b, c, d, e, b, f\rangle^4, \langle a, e, b, c, d, b, f\rangle^3\ ]$

If we apply the algorithm, it goes as follow:

1. $T_L = \{a, b, c, d, e, f\}$

   T_L is a set of activities that appears in the log L

2. $T_I = a$
3. $T_O = f$

   a is the start activity and f is the end.

4. X_L = { ({a}, {b}), ({a}, {e}), ({b}, {c}), ({b}, {f}), ({c}, {d}),

   ({d}, {b}), ({e}, {f }), ({a, d}, {b}), ({b}, {c, f}) }

   The Alpha algorithm scans the event for particular patterns if activity a is followed by b, but b is never followed by a, then it is assumed that there is a causal dependency between a and b. There are four log-based ordering relations that aim to extract relevant pattern in the log:

   - $a >_L b$ if and only if there is a trace $\sigma = \langle t_1, t_2, t_3, ..., t_n \rangle$ and $i \in \{1, ..., n-1\}$ such that $\sigma \in L$ and $t_i = a$ and $t_i + 1 = b$
   - $a \rightarrow_L b$ if and only if $a >_L b$ and $b \not>_L a$
   - $a \#_L b$ if and only if $a \not>_L b$ and $b \not>_L a$
   - $a ||_L b$ if and only if $a >_L b$ and $b >_L a$

   Using the patterns we have results as describe in the Table.3.1 and we took all activities that $a \rightarrow_L b$.

Table 3.1: Footprint of L_5 [36]

	a	b	c	d	e	f
a	#	→	#	#	→	#
b	←	#	→	←	||	→
c	#	←	#	→	||	#
d	#	→	←	#	||	#
e	←	||	||	||	#	→
f	#	←	#	#	←	#

5. Y_L = { ({a}, {e}), ({c}, {d}), ({e}, {f }), ({a, d}, {b}), ({b}, {c, f }) }

   We need to filter elements of X_L, we chose the "maximal pairs" (A, B) for any pair (A, B) $\in X_L$, nonempty set $A' \subseteq A$, and nonempty set $B' \subseteq B$, it is implied that $(A', B') \in XL$.

6. P_L = { $p_{(\{a\},\{e\})}$, $p_{(\{c\}, \{d\})}$, $p_{(\{e\}, \{f\})}$, $p_{(\{a, d\}, \{b\})}$, $p_{(\{b\}, \{c, f\})}$, i_L, o_L }

   For each element of (A, B) $\in Y_L$ should corresponds to a place $p_{(A,B)}$, connecting transition A to transition B. Moreover, since the objective of this algorithm to create an WF-NET it also must have a unique source and sink place of i_L and o_L.

7. F_L = { (a, $p_{(\{a\},\{e\})}$), ($p_{(\{a\},\{e\})}$, e), (c, $p_{(\{c\}, \{d\})}$), ($p_{(\{c\}, \{d\})}$, d),

   (e, $p_{(\{e\}, \{f\})}$), ($p_{(\{e\}, \{f\})}$, f), (a, $p_{(\{a, d\}, \{b\})}$), (d, $p_{(\{a, d\}, \{b\})}$),

   ($p_{(\{a, d\}, \{b\})}$, b), (b, $p_{(\{b\}, \{c, f\})}$), ($p_{(\{b\}, \{c, f\})}$, c),

   ($p_{(\{b\}, \{c, f\})}$, f), (i_L, a), (f, o_L)}

8. $\alpha(L) = (P_L, T_L, F_L)$

   After we define places and transitions, we just need to create the arcs of the WF-Net, each place in $p_{(A,B)}$ have A as input nodes and B as output nodes, starting from i_L as an input place and ended in o_L. Finally, we have $\alpha(L) = (P_L, T_L, F_L)$ that describes the behavior seen in event log L. The WF-NET of this log can be seen at Fig.3.3.

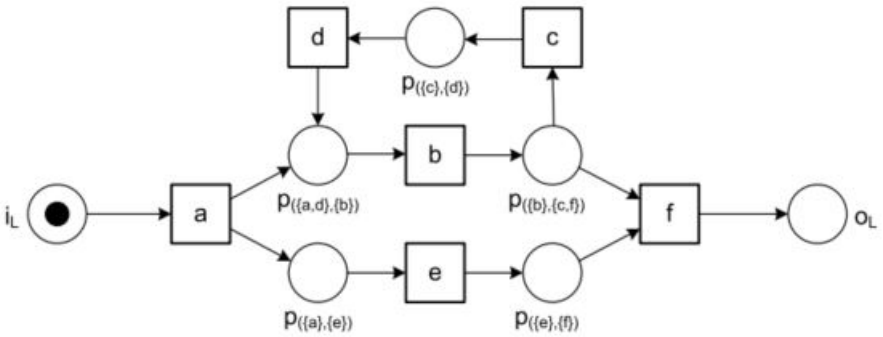


Figure 3.3: WF-NET L_5 [36]

3.2.2 Heuristic Miner

Heuristic miner is one of the advance process discovery techniques in generating a process model [36]. The algorithms take consideration of frequencies of events and sequences when creating the model. The basic principle of this algorithm is that infrequent path should not be incorporated into the model, the miner also able to distinguish between noise and incomplete behaviour of an event log. The algorithms consist of the main heuristic miner (Definition.10). Mining dependency graph (Definition.11) is a subset of the main heuristic miner, to explore a dependency matrix between activities. Short loop (Definition.12) is useful to detect loop in an event log. Lastly, split and join between activities can be identify using split/join (Definition.13).

Definition 10 (Heuristic Miner [44]). *Let T be set of activities.* $\sigma \in T^*$ *is an event trace, an arbitrary sequence of activity identifiers.* $W \subseteq T^*$ *is an event log, multiset (bag) of event trace. Let* $a, b \in T$ *:*

1. $a >_w b$ *iff there is a trace* $\sigma = t_1 t_2 t_3 ... t_n$ *and* $i \in \{1, ..., n-1\}$ *such that* $\sigma \in W$ *and* $t_i = a$ *and* $t_{i+1} = b$,
2. $a \rightarrow_w b$ *iff* $a >_w b$ *and* $b \not>_w a$,
3. $a \#_w b$ *iff* $a \not>_w b$ *and* $b \not>_w a$,
4. $a || _w b$ *iff* $a >_w b$ *and* $b >_w a$,
5. $a >>_w b$ *iff there is a trace* $\sigma = t_1 t_2 t_3 ... t_n$ *and* $i \in \{1, ..., n-2\}$ *such that* $\sigma \in W$ *and* $t_i = a$ *and* $t_{i+1} = b$ *and* $t_{i+2} = a$,
6. $a >>>_w b$ *iff there is a trace* $\sigma = t_1 t_2 t_3 ... t_n$ *and* $i < j$ *and* $i, j \in \{1, ..., n\}$ *such that* $\sigma \in W$ *and* $t_i = a$ *and* $t_{i+1} = b$.

Using the standardisation of XES (Fig.2.3), W is a *Log*, σ is *Trace*, and T is an *attribute* of an *event*. The step 1 is getting the first and second ordered activities that denotes in $t_i = a$ and $t_{i+1} = b$. The step 2, is to find causality between activity a and b, if a directly followed by b and not the other way around, hence there is direct dependency b to a. In the step 3, if a is not directly followed by b and vice-versa, it means there is no relationship between a and b. Next, in the step 4, if a is directly followed by b and vice-versa, thus relationship between a and b is a parallel / concurrent relationship. Step 5 is to identify short loop, for example activity a goes to activity b and return to activity a. Lastly, step 6 is used to identify long range relationship between activity a and b.

Definition 11 (Mining dependency graph [44]). *Let W be an event log over T and* $a, b \in T$. *While* $|a >_w b|$ *is the number of times* $a >_w b$ *occurs in W, then:* $a \Rightarrow_w b = (\frac{|a >_w b| - |b >_w a|}{|a >_w b| + |b >_w a| + 1})$

The dependency graph is the starting point for Heuristic Miner. The graph acts as a frequency based metric that is useful to indicate the dependency between two activities. A high $a \Rightarrow_w b$ suggests that there is a dependency relation between activity a and b. Heuristic miner has the capability to detect noise, to support this feature there are three threshold parameters: (1) the *dependency threshold*, threshold to filter events using dependency measurement.(2) The *positive observation threshold*, a threshold to filter events using frequency of occurrences.(3) The *relative to best threshold*, a threshold to filter events by comparing the dependency measurement of an event relative to the best dependency measurement in the event log.

Definition 12 (Mining short loop [44]). *Let W be an event log over T, and $a, b \in T$. The $|a >_w b|$ is the number of times $a >_w b$ occurs in W, and $|a >>_w b|$ is the number of times $a >>_w b$ occurs in W.*

1. $a \Rightarrow_w a = (\frac{|a>_w a|}{|a>_w a|+1})$
2. $a \Rightarrow_{2w} b = (\frac{|a>>_w b|+|b>>_w a|}{|a>>_w b|+|b>>_w a|+1})$

Heuristic miner is able to facilitate a short loop of events. (1) is used to calculate dependency measurement of a recursive loop, and (2) is to calculate dependency of a short loop, for example activity a goes to activity b and return to activity a.

Definition 13 (Mining AND/XOR-split/join and non-observable tasks [44]). *Let W be an event log over $T, a, b, c \in T$, and b and c are in depending relation with a. Then: $a \Rightarrow_w b \wedge c = (\frac{|b>_w c|+|c>_w b|}{|a>_w b|+|a>_w c|+1})$*

Using this formulation, we can identify split and join relationship between activities. In the research they using threshold value 0.1 to separate AND-relation with XOR-relation. $|a >_w b| + |a >_w c|$ indicates the number of positive observations and $|b >_w c| + |c >_w b|$ indicates the number of times b and c appears directly after each other.

3.2.3 Default Miner

Default miner is a simple and straight forward algorithm to correlate relationship between *activities* and *originators* (entity that execute the tasks).

Definition 14 (Default Miner [33]). *Let L be an event log, T be a set of activities, and c is possible event sequence, $c = (e_0 e_1, \ldots) \in L$, P a set of originators (i.e., persons, resources, or agents). $E = TxP$ is the set of possible events, combination of an activity and an originator.*

1. *For each* $t \in T, O_t = \{\pi_p(e) | \exists_{c \in L} e \in c \wedge \pi_t(e) = t\}$
2. $A_s = \{(t, O_t | t \in T)\}$

In (1) organisational entity (O_t) denotes for an activity (t) has originator that executed the task (t), while (2) A_s denotes relationship between organisational entity (O_t) with activities (t). Organisational entity do not disjoint, because an originator can perform activity more than one, thus relationship between organisational units with activities are many-to-many. To underline which organisational unit execute which activity, default miner also utilise *Metrics based on joint activities*, two dimensional matrix of how frequent an originator execute an activity in the log.

3.2.4 Disco and PromLite

As alternatives for comparing the results, we used other process mining tools, such as Fluxicon-Disco[1] and PromLite 1.2[2]. Both tools quite easy to use and accessible, PromLite can be used freely, and Disco can be accessed using an academic license. The prom lite has multiple plugins for multiple perspectives, so it is quite versatile to test a variety of process mining algorithms and perspectives. While disco only limited to certain control flow and data/time perspectives. Although it is simple, disco provides users with user friendly interface and nice graphical notation. Both tools can provide a comparable result that useful in validating our research, since our research is combining multiple perspectives of process mining, and the current standard does not provide a tool for that purpose, thus we can only limit our validation to a certain perspective of process mining. In regards to these tools, we only used interactive data-aware heuristic miner plugin from prom lite, and standard graph from disco, that similar to the dependency graph. Figures.3.4, 3.5, 3.6 are the user interfaces from promlite and disco.

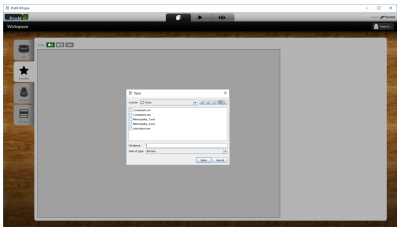


Figure 3.4: PromLite main user interface

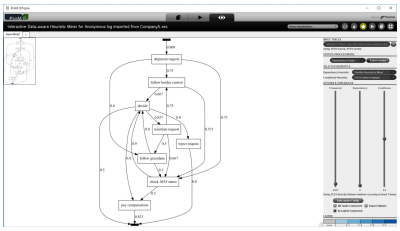


Figure 3.5: PromLite result example

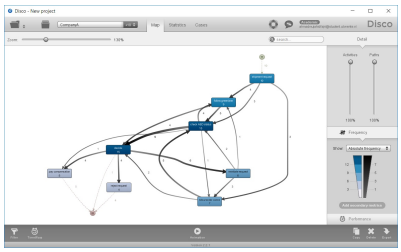


Figure 3.6: Disco main user interface

[1] https://fluxicon.com/disco/
[2] http://login.webofknowledge.com

4

EA mining

This chapter will discuss about the research's artefact: EA mining. First we will talk about EA mining overview and workflows. After that we will discuss log structure, one of the EA mining artefacts. It consists of: the log structure, the guideline to populate the log structure, relevant fields that can be used in the log structure. Then, we will talk about EA mining conversion methods, steps that must be taken, and detail explanation of each steps.

4.1. EA mining overview

EA mining consists of two artefacts, log structure and conversion methods. Log structure is defined how event logs should consist of, so it can be used for EA mining purposes. Next, conversion method is definition of methods to process the event log to an EA model. Figure.4.1 illustrates the overview of EA mining, we colour-coded the figure, the blue parts are elements that are used for creating an event log, it is not part of the research and we put there to illustrate how the event logs are produced. The green parts are the artefacts of this research it consists of an event log that follows the log structure that we proposed and conversion method to process the log into EA models (orange parts).

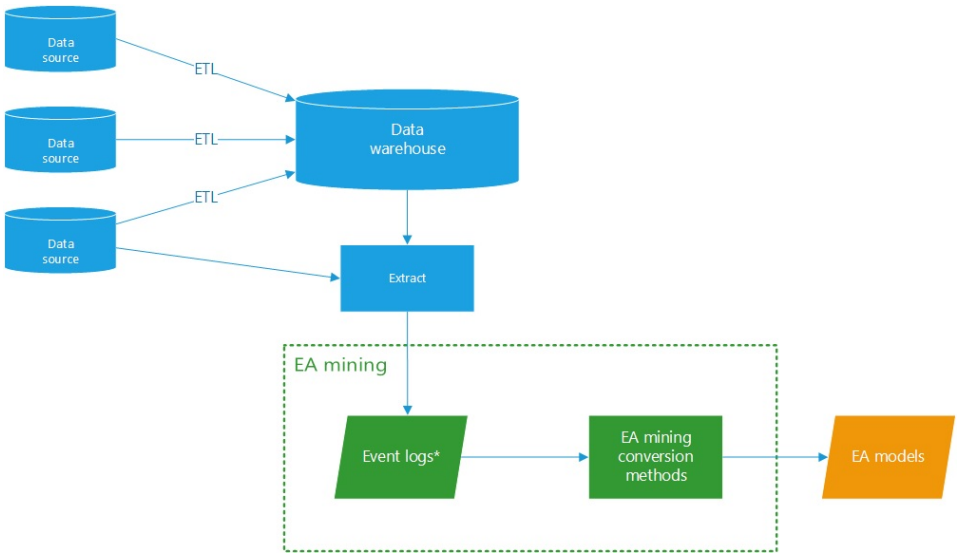


Figure 4.1: EA mining workflow

Note: Event log* refers to event logs in the log structure format

4.1.1 Log Structure

An event log can be used to generate ecosystem or surrounding components in the entire enterprise, event log is a digital footprint that generated from systems' activities, these footprints not only contain about the information of the system itself, it can also contain information of its surrounding systems. Typically, a system can have multiple cores or supporting systems that can describe an entire ecosystem of the processes. For example, an e-commerce B2C e-Shop can have: web-portal for browsing products and place orders, payment systems to finalise payment, logistic systems to send products, and financial systems to record sales volume. Each step in the whole systems can be recorded as an event and can be put into an event log with additional informative attributes, such as the name of the system, who execute the event, time its executed, and also additional performance measurement, such as completion time, a response time of each event. Using this event log, we can observe how the process works and how is the interaction between systems. In the end, we can get a general picture of the whole processes with its ecosystem, its corresponding systems, and its stakeholders.

Van der Aalst [36] mentioned how to produce the log (Section.2.2.1.4), an event log can comprise from multiple data sources from multiple systems, this concept can be supported by studies from [13]. Farwick et al. [13] describe a list of productive systems that contains relevant EA information. Other studies [9, 22, 42] also describes productive systems that available for EA conversion, with more definitive examples. Thus, the list of available systems encourages the idea of the log is created from multiple data sources. It also adds additional features that were not covered in other studies. In [9, 22, 42] conversion process limited to a certain system, tools, and work processes. While using event log, we can combine all possible data sources into one cohesive log, that encompass all systems, which in the end can be used to describe ecosystems of the process and interaction between systems and stakeholders. This idea can give us a general perspective of an enterprise which align with the purpose of enterprise architecture.

In this study we used metamodel of XES as our log structure (Fig.2.3). An event log will be consists of *Trace* and each *Trace* consists of an *Event*. *Trace* and *Event* can describe with *Attributes* that consist of key and value, with multiple options of data types. Data types itself can be chosen from *String, Date, Integer, Float, Boolean*.

4.1.2 Possible relevant fields

Log in the beginning produced from "raw" hidden in varied data sources, the data sources itself can be a simple flat file, an Excel spreadsheet, a transaction log, a database table. One of the assumptions that need to be adhere when creating a log file is that data itself is not to be expected provided in one single, well-structured data source. Usually when creating a log file, we must gather from multiple data source, and efforts are needed to gather the relevant data [36]. In previous research of automated EA model documentation, we found that data source in creating an EA model is limited to certain system, tools, and software. Such as [22] using network scanner, [9] using ESB and [42] using PAIS. However, in creating an event log we can get over these boundaries, since the log can be created from multiple data source.

As we discussed before, there are exists at least six possible data source available from productive systems for automated EA documentation [13]. *Network monitor and scanners*, a tool to gather network activities using network scanners and sensors. *Configuration management database*, a tool to collect operational data in an organisation that comply with ITIL. *Project portfolio management tools*, a tool for managing project portfolio. *Enterprise service bus*, central mediating entities for inter-application communication. *Change management tools*, a tool to optimise implementation of changes in the IT-landscape. Lastly, *license management tools*, a tool to manage acquired software licenses. With these multiple options, there are possibility to produce more cohesive log that can be used for automated EA documentation. Thus, using this assumption we propose possible log structure that can be used for automated EA documentation in the future. The following table (Table.4.1) is compilation of data source that available, with possible fields in the respective data source with possible translation in Archimate elements and layer. This table is produced based on literature

review that we conducted before (Section.2.2.2.1). However, not all the fields can be easily picked up as events' attributes, it must have some key that bound from one data source to other. In the example of e-commerce previously, we can set *order number* as an intermediary key between systems. While in the table (Table.4.1) it somehow difficult to connect network monitoring fields to other systems without sufficient context and key.

Table 4.1: Possible log structure

data source	**Source (Fields)**	**Target (Archimate's Elements)**	**Layer**
Business process [3, 7, 30, 42]	Activity	Business Process	Business
Business process [3, 7, 30, 42]	Resource	Business Actor, Application Component	Business, Application
CMDB [1, 2]	Product	Device / System Software	Technology
CMDB [1, 2]	Owner	Business Actor	Business
Network monitoring [22]	User Account	Business Actor	Business
Network monitoring [22]	Application protocol	Infrastructure interface	Technology
Network monitoring [22]	Application server	System software (end-point)	Technology
Network monitoring [22]	Application Client	Application component	Technology
Network monitoring [22]	Operating System	System software (OS)	Technology
Network monitoring [22]	IP, MAC Address	Device	Technology
Network monitoring [22]	A range of IP address	Network	Technology
ESB [9]	Party	Business Actor	Business
ESB [9]	data type	Business Object	Business
ESB [9]	adapter	Representation	Business
ESB [9]	data type	Data Object	Application
ESB [9]	enterprise service interface	Application Interface	Application
ESB [9]	software component, software product	Application Component	Application
ESB [9]	sofware configuration	Application Collaboration	Application
ESB [9]	computer system	Node	Technology
ESB [9]	system landscape directory	System Software	Technology
ESB [9]	outbound and inbound interface	Communication Path	Technology

4.1.3 Basic Log Structure for EA Mining

Based on the possible relevant fields (Section.4.1.2), there are multiple fields that can be used as base log, which in the end to be used for EA mining conversion. However, not all the fields can be easily used for the EA mining purposes. The basic form of the log structure is consists of Trace, that identify one workflow and cluster group of events of that one particular workflow. Each traces contains one or multiple events. Events contains activities that an enterprises have. Each events contains additional information that are recorded as attributes. The baseline of EA mining is control-flow (Section.2.2.3.2), which means that all the information of the log wanted to express must be connected to events. For this research, we propose attributes that all the system might have. The attributes are *Activity, Timestamp, Login, AppName,* and *ResponseTime*. *Activity* refers to the name of the event. *Timestamp* is the date that the event occurs, all event logs contains the activity and timestamp [3, 7, 30, 42]. *Login* refers to stakeholders that execute the event, it can be seen that this information is accessible from systems [1–3, 7, 9, 22, 30, 42]. *AppName* refers to the application name that the event executed from, it is also possible to access this information since an event is usually generated from a certain system, this information is also accessible from systems [9, 22]. Lastly *ResponseTime*, it refers to the elapsed time between an inquiry on a system and the response to that inquiry. This is an additional fields that used for the purpose of explaning that not only log useful for generating EA model, but also it can add new dimension of the EA model by further analysis the model. The log structure is illustrated in the Figure.4.2.

Figure 4.2: Log structure

4.1.4 Log Guideline

We also propose a simple event log creation guideline. The guideline consists of four steps. *"Identify a business model", "Identify Activities, Key, and Workflow", "Identify Traces and Events"*, and *"Collecting relevant attributes and create a log"* (Fig.4.3). Each step is further explained in the following sections.

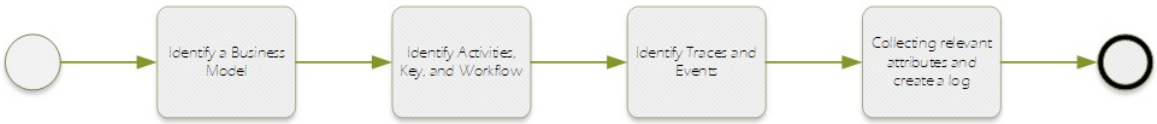


Figure 4.3: Event log guideline

4.1.4.1 Identify Business Model

In a big enterprise we can find multiple business models that the enterprise does, for example Amazon has multiple business models, such as Amazon Marketplace, Amazon Books, Amazon Web Services, Amazon Gaming, Amazon Music and Videos, Amazon Prime, Amazon Fire Products [15]. In this step we need to identify which business model that we want to observe its architecture.

4.1.4.2 Identify Activities, Key, and Workflow

After we identify which business model that we want to focused on, the next step is we need identify activities of that business model. For example, in a typical e-commerce business model it is consists of activities: browse and place order, payment, shipment, after sales [34]. After that we need to identify the key to link between those activities. Next, we need to determine the workflow of those activities, which are the starting and end points, and sequence of the flow. The workflow itself might be consists of combination of activities and its variation can be differ from one workflow to others.

4.1.4.3 Identify Traces and Events

The next step is identification of cases and events. Continuing from the previous step, each workflow might be differ from one to another, thus each workflow sequence is recorded as a *Trace*. And the activity that we discussed in the previous step is recorded as an *Event*. *Events* may have arbitrary numbers to distinguish each from another. In *Traces* we used *Key* as a unique identifier to group *Events*.

4.1.4.4 Collecting relevant fields and create a log

The final step is collecting relevant fields and create a log. In collecting fields there are multiple fields and data sources that are available depending on the data source's media and format. In the e-commerce system we can have multiple system, for example a website that save its data in database, while a third party payment system might provide the enterprise with a csv file and shipping company with APIs and an internal finance system save its data

within ERP system. There are variety of options of data sources and fields. The technicality of how to identify and to extract beyond the scope of this study. The guideline only emphasise which fields that we want to observe and save those fields into a log as attributes of an event. The attributes might be a name of the event's system, which user access the event, what time does the event occurs.

4.2. EA mining conversion method

In an example of a typical e-commerce system, which consists of web-portal, payment, logistic, and financial system. In order to complete one order from a customer, the e-commerce system need multiple systems. The order will flow through the system from placing the order, payment, product shipment, and bookkeeping the sale. This workflow can be captured using process mining control-flow perspective algorithm. The control-flow perspective enabled a user to discover the behaviour of processes through the workflow. Although the workflow emphasise the process (in the previous example of e-commerce, the order) but the order itself not only contain the order information but also other information, such as *activity* to describe the event name, *resource* who executes the event, *time* the event conducted, and measurement of the event (response time, completion time). Hence, this information can be used to describe the every components that related to the workflow, such as systems, and stakeholders. Based from those idea, we propose EA mining, steps to process an event log to produce an EA model. EA mining consists of three algorithms, *business process discovery, workflow related elements discovery*, and *EA analysis functions*. *Business process discovery* is an algorithm to discover workflow in a log in order to observe business process behaviour within an enterprise. *Workflow related elements discovery* is an algorithm to discover elements that related to the workflow, lastly *EA analysis functions* is an algorithm to analysis event measurement.

The EA mining is executed in the sequential order, first of all is *business process discovery*, the core algorithms of the EA mining, since we need to discover a workflow first before discovering surrounding elements that attached to the workflow. Next step is *workflow related elements discovery*, to discover elements that related to the workflow. Lastly, *EA analysis function* to analyse measurement of the workflow. The illustration of these steps can be seen at Fig.4.4.

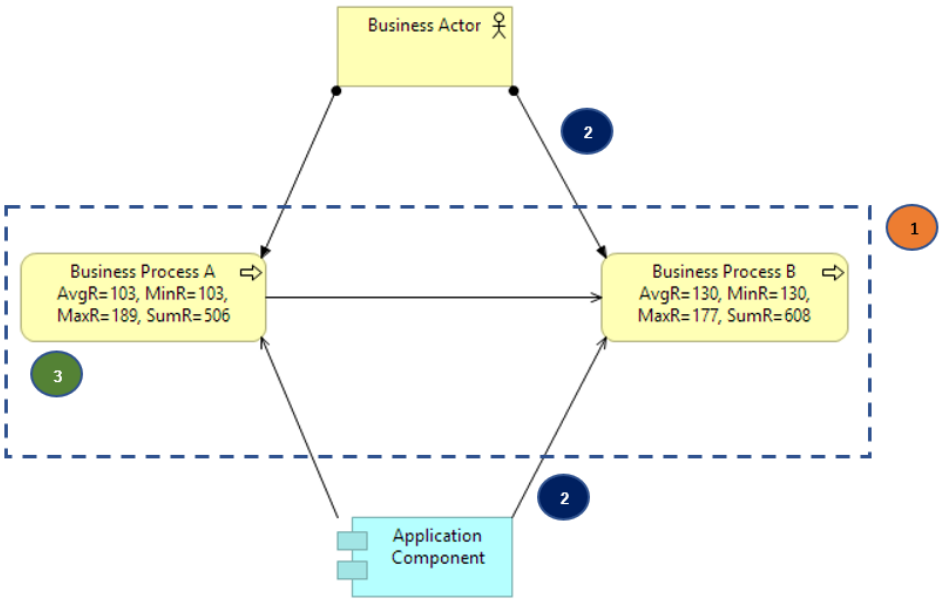


Figure 4.4: EA mining algorithms overview

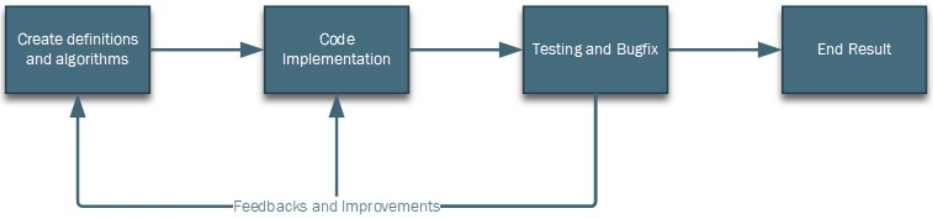


Figure 4.5: EA mining development procedure

In developing of our algorithms, first of all, we create definitions and algorithms. Then, we implement the definition and algorithms into programming language/code. Next, we test our prototype and bug-fix. The testing procedure limited to functional testing, to make sure that the code produces an expected result based on the definition and algorithms. We iterate the process if we find feedbacks and improvements for previous processes, Lastly, if the testing produces expected results as intended in the definition and algorithms, then we end the development process. The illustration for this process van be seen at Fig.4.5. The development procedure is explained more detail in three chapters, the definition is described in this chapter (Chapter.4), while algorithms and implementation in the implementation chapter (Chapter.5), and the testing and end result is explained in the validation chapter (Chapter.6).

4.2.1 EA mining definition

4.2.1.1 Step 1: Business Process Discovery

Process mining has multiple perspective to analyse the behaviour of processes in systems. One of the perspective is control-flow perspective, with goals to find a good characterisation of all possible paths of processes. Usually this perspective expressed in terms of a Petri net or other notation (i.e, BPMN, UML activity diagram) [36]. In this research, we want to incorporate one of the control-flow perspective algorithm to discover business processes that resides in logs of systems of an enterprise. We called this mechanism as *business process discovery* algorithm, to discover business process that resides in logs, that enable us to observe business process behaviour of systems. This algorithm will be the backbone of our EA mining and as a foundation for extending other EA mining algorithms. We used heuristic miner (Definition.10) as a baseline for the algorithm, and the algorithm result limited to Archimate. We chose Heuristic miner since it is enable us to discover processes and its concurrences and also its capabilities to deal with noise (low frequency activities) and infrequent/incomplete activities.

Heuristic miner produces dependency graph (Definition.15) as its process model [44], the dependency graph represents all the dependencies found in the log. Fig.4.6 is an example of dependency graph that resulted from a heuristic miner processing. As can be seen at the figure, it has rectangle to represent an activity with its frequency occurrences, the arc represent dependency path between activities, the arc also has label that represent dependency factor and its frequency. We proposed the conversion pattern in the literature study chapter (Section.2.2.4) and the result of *business process discovery* algorithm is similar to the dependency graph, and instead of rectangle as an activity, we substitute it with *business process* element, and we convert arcs to *serving* relationship. Example of the converted graph can be seen at Fig.4.7.

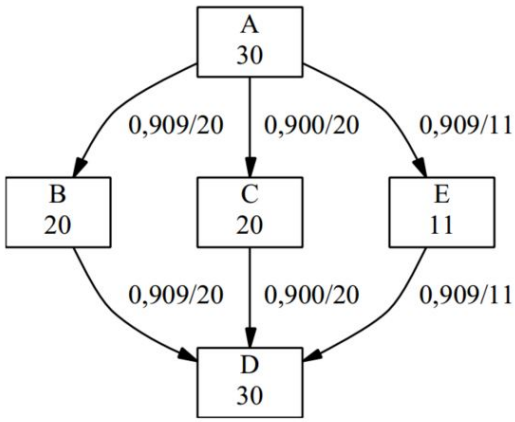


Figure 4.6: Example dependency graph [44]

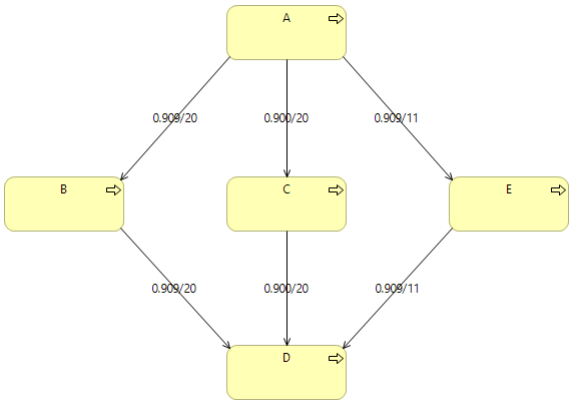


Figure 4.7: Example of converted dependency graph to Archimate

Definition 15 (Dependency Graph [4]). *Given a set of activities V and a log of executions L of the same process, a directed graph G_{VL} is a dependency graph if there exists a path from activity u to activity v in G_{VL} if and only if v depends on u.*

The input for this algorithm is the log structure (Section.4.1.1), which complies to the definition of simple event log (Definition.4). The simple event log is subset of the log structure, since the input for this algorithm focuses only on activity name of events. Thus we defined *business process discovery* as follows:

Definition 16 (Business Process Discovery). *Let L be an event log over activities of T, with trace of σ.*

Step 1. Identify activities in the log
$T_L = \{t \in T \mid \exists_{\sigma \in L}\, t \in \sigma\}$.
The first step of the algorithm is to identify activities in the event log. Assuming that we have an event log of $L_4 = [(a,c,d)^{45}, (b,c,d)^{42}, (a,c,e)^{38}, (b,c,e)^{22}]$, (a,c,d) is a trace, while "a" in (a,c,d) denotes an activity (attribute) of an event. By applying the step 1, activities will be $T_L = (a,b,c,d,e)$.

Step 2. Constructing activities sequences
In step 2, the algorithm will produce activities sequences of a trace in the event log, using the following sub-step:

a. $a >_L b$ *iff there is a trace* $\sigma = t_1 t_2 t_3 ... t_n$ *and* $i \in \{1, ..., n-1\}$ *such that* $\sigma \in L$ *and* $t_i = a$ *and* $t_{i+1} = b$,

b. $a \rightarrow_L b$ *iff* $a >_L b$ *and* $b \not>_L a$,

c. $a \#_L b$ *iff* $a \not>_L b$ *and* $b \not>_w a$,

d. $a ||_L b$ *iff* $a >_L b$ *and* $b >_L a$,

By applying Step 2a, the algorithm will determine which activities appeared in sequence, for example of L_4, it will produce $(a >_L c, c >_L d, b >_L c, c >_L d, a >_L c, c >_L e, b >_L c, c >_L e)$. $a >_L c$ denotes that activity c will follow activity a. Relation $\rightarrow_L$ denotes a direct dependency relation, using the definition of Step 2b, we identified direct dependency relationship between activities. Relation $||_L$ denotes concurrent behaviour, or potential parallelism between activities. We used Step 2d, to identify this behaviour. While $\#_L$ denotes two activities never follow each other directly. The summarization of all behaviour activities of L_4 can be seen at Table.4.2.

Step 3: Create a frequency matrix
We count occurrences for each sequence of activities $a \rightarrow_L b$ and produce a frequency matrix. An example of the matrix of L_4 can be seen at Table.4.3.

Step 4: Create a dependency matrix
Using formulation of $a \Rightarrow_L b = (\frac{|a>_L b|-|b>_L a|}{|a>_L b|+|b>_L a|+1})$ (Definition.11), we can generate dependency matrix of each activities, the dependency matrix for L_4can be seen at Table.4.4.

Step 5: Create a result matrix
To simplify the result of the algorithm, we combine all the information from step 2-4 and produce a matrix that consists of sequence of activities with columns source and target, with its frequency and dependency factor. Table.4.5 is an example of result table using the log of L_4 and if the result converted into a converted dependency graph it becomes Fig.4.8.

Table 4.2: Footprint of L_4

	a	b	c	d	e
a	#	#	→	#	#
b	#	#	→	#	#
c	←	←	#	→	→
d	#	#	←	#	#
e	#	#	←	#	#

Table 4.3: Frequency matrix of L_4

	a	b	c	d	e
a	-	-	83	-	-
b	-	-	64	-	-
c	-	-	-	87	60
d	-	-	-	-	-
e	-	-	-	-	-

Table 4.4: Dependency matrix of L_4

	a	b	c	d	e
a	-	-	0.99	-	-
b	-	-	0.98	-	-
c	-	-	-	0.99	0.98
d	-	-	-	-	-
e	-	-	-	-	-

Table 4.5: Finalise table L_4

source	target	frequency	dependency
a	c	83	0.99
b	c	64	0.98
c	d	87	0.99
c	e	60	0.98

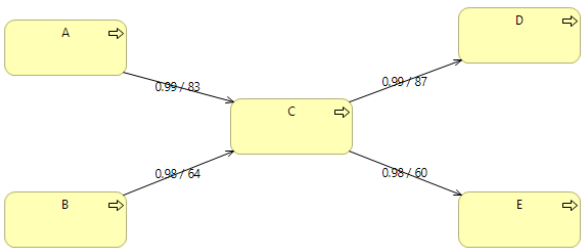


Figure 4.8: EA model conversion of L_4

4.2.1.2 Step 2: Workflow related elements discovery

Organisational perspective in process mining is a mechanism to discover the hidden information regarding resources in a log, which actors (i.e. people, system, roles, and departments) are involved and relationships between them. Van Der Aalst and Song [39] able to derived interaction between people when executing activities in event logs and generate *sociograms*. They propose matrices to derive the sociograms. One of the metric that relate to our study is *metric based on joint activities*. In the metric, it shows which activities that an individual does in event logs. Using assumptions that people doing similar activities have stronger relationship than people doing completely different activities, and using the assumption to

discover the quality of relationships based on how often people do the same activity. Using the metric we proposed the *workflow related elements discovery*, to identify *people* that responsible to execute an *event*. *People* can be translate in Archimate into *business actor*, *event* to *business process*. To relate between those two elements Archimate provides *assignment, serving, triggering, flow,* and *association* [18, p.125]. However to simplify the research we chose *assignment* to represent a responsibility of an entity to conduct a certain task. In addition, although the metric originally focuses on the relationship between activities and performer (people that executes the activities), however we think that the metric can also be used to check the relationship between systems and activities. Reasoning behind this simply not only people that can execute a task, it also possible that certain roles execute the task, or systems that execute the task. Thus, we broad our definition of performer into an actor (i.e. people, system, roles, and departments). Thus, in the end the metric not only limited to people but also to systems or business roles. *System* can be represented in Archimate as active structure elements such as *application component*, or *system software*. Since we want to limit our research we only focus on application layer. Hence, the system is represented as *application component*. Archimate provides *realization, serving, triggering, flow*, and *association* to relate between *application components* and *business process* [18, p.126]. And we chose *serving* to represent a system that serves business process by conducting certain task. Overall, the algorithm will produce *business actor* with *assignment* to *business process* and *application component* to *serve* a *business process*.

Definition 17 (Metrics Based on Joint Activities [39]).
Let L be a log. For $p_1 \in P, a_1 \in A$, *and* $c = \{c_0, c_1, ...\} \in L$:

1. $p_1 \vartriangle_c a_1 = \Sigma_{0 \le i < |c|} \begin{cases} 1 & if\ \pi_a(c_i) = a_1 \wedge \pi_p(c_i) = p_1 \\ 0 & otherwise \end{cases}$
2. $p_1 \vartriangle_L a_1 = \Sigma_{c \in L} p_1 \vartriangle_c a_1$

Note that $\vartriangle$ denotes a matrix with rows P and columns A. While A is a set of activities (i.e., atomic workflow/process objects), P is set of actors (i.e. people, system, roles, and departments), and C are cases or traces. Moreover, $\pi_a(c) = a$, and $\pi_p(c) = p$ denotes an operation of some event $e = (a, p)$. In (1) it means that in each case of the log, it flag 1 for a case that an actor executes activities and 0 if not. In (2) denotes that combining all flag that had been calculated previously in (1). Using the metric we can identify which person or system that responsible to conduct certain service to which activity and we can also quantify it, to see how frequent that interaction occurs. Table.4.6 is an example of the metric with actor as a person.

Table 4.6: Metrics based on joint activities example

Actor	Activity A	Activity B	Activity C
Peter	1	1	0
Quagmire	0	1	0
Cleaveland	1	0	1
Joe	0	0	1

4.2.1.3 Step 3: Analysis Function

The last step of EA mining is to conduct analysis function of an EA model. The objective of this step is to prove that event log not only contain necessary information related to general overview of systems in an enterprise, but it can also contain necessary information that can be used for analysis. To achieve this task, in this step we used simple mathematical operations, such as summation, mean, minimum and maximum to analyse *Response Time* attribute of an event.

Definition 18 (Analysis function).
Let $\mathscr{L}$ be an event log. $\mathscr{E}$ is set events in the log. $\#_{activity}(e)$ be an activity associated to the event e and $\#_{respTime}(e)$ is a response time that associated to the event e. $ACT = \{act_1, act_2, ..., act_m\}$ is a set activities of the log. $RPA_i = \{rpa_{i1}, rpa_{i2}, ..., rpa_{im}\}$ is a set of ResponseTime of an $activity_i$. $\forall_{act_i} \in ACT$ and $\forall_{rpa_{ij}} \in RPA_i$, hence:

Summation
$sum\_rpa_i = \Sigma_{j=1}^{n} rpa_{ij}$

Mean
$mean\_rpa_i = \frac{Sum\_rpa_i}{n}$

Minimum
$min\_rpa_i = min\{rpa_{ij}\}_{j=1}^{n}$

Maximum
$max\_rpa_i = max\{rpa_{ij}\}_{j=1}^{n}$

5

Implementation

This chapter will discuss about code implementations of the EA mining definitions. First, we will talk about the implementation mapping from EA mining definition to algorithms, next we talk about categorisation of algorithms, algorithms that were used from definition and algorithms that used for prototyping. After that we will discuss in detail of each algorithms. Next, we will talk about Archi, its metamodel, and the algorithms that were used to create Archi files. Lastly, we will discuss the prototype of the implementation, its user interface, and its workflows.

Figure 5.1: Implementation procedures

5.1. Code Implementations

After we defined the EA mining definitions (Section.4.2.1) we can implement the algorithms. Each of the definitions will be implemented into separate algorithms. However, for workflow related element discovery we split the algorithms into two parts, one for the business actor and the other for application component. We also defined supporting algorithms that useful for prototyping the implementation. The implementation procedures can be seen at Fig.5.1, we colour coded the figure into three categories. The green ones are to identify definitions

(Section.4.2.1), the purple ones are the implementation of the definition. Lastly, the orange ones are for supporting algorithms. Each of the algorithms will be discussed in detail in the following sections.

5.1.1 Business Process Discovery Algorithm

5.1.1.1 Read log file algorithm

Algorithm 1 Read log file

```
1: procedure ReadLogFile(logPath)
2:     parser = TextParser(logPath)
3:     while parser is not at the end of the file do
4:         arrFields = read current fields from parser
5:         data = new row of logData
6:         data.Id = arrFields[0]
7:         data.Activity = arrFields[2]
8:         insert data into logData
9:     return logData
```

The first implementation of the business process discovery algorithm is to read the data from a log file. The *ReadLogFile* algorithm (algorithm.1) requires the *logPath*, Microsoft Windows path, that directed to the location of the log file. This algorithm produces a *logData*, a table that structured refers to the metamodel of the log file (Fig.4.2). The table consists of *TraceId* that cluster events based on case, *EventId* unique identifier of an event, *Activity* the name of an activity, *Resource* a person who executes an activity, *Timestamp* the time that an activity executed, *AppService* the application service of that accommodates an activity, *AppName* an application name that facilitates an activity. In the beginning, the algorithm creates a text parser to read the log file, since the file is structured as a csv file then the parser needs to consider the delimiter of the file, in this implementation we used "," (comma-delimiter). After that, the parser will loop for each line of the file and put it on the *arrField* array of the string data type. The next step, the algorithm will insert each array fields as a new row of the table *logData*. The fields that needed for the algorithm.2 only *TraceId* and *Activity*. Finally, the *ReadLogFile* algorithm will return *logData*.

5.1.1.2 Business Process Discovery Algorithm

In the business process discovery algorithm (algorithm.2), first of all, the algorithm reads the log file using ReadLogFile algorithm (algorithm.1) and produces *logData* table, currently the log data table consists only *TraceId* and *Activity*. Next, the algorithm creates sequence of processes and occurrences of each sequences. The algorithm reads each row of *logData* and identify cluster for each Trace/Case using *TraceId*. In step.10 and 11 the algorithm will identify *id* for a new cluster and the beginning activity for each clusters (*actA*). If the $id = row.id$ means that current row still in the same cluster, then the algorithm will get the next activity (*actB*) and create a *key* of $"actA \rightarrow actB"$ to identify sequence of process (step.15). After that, in step.16 the algorithm will check to a dictionary, whether the key has already existed in the dictionary. if it existed, then save the occurrence, if not existed then create the dictionary (if the dictionary has not been initialised before) and add new entry in the dictionary with key and initial occurrence. The loop process continue until all the records in the logData exhausted.

In the next block of code, the algorithm will create a frequency table. Looping all records in the dictionary that were created before. In this block of code, the algorithm will save the current dictionary entry into *dictItem* (step.25) and create a new row for an entry into result table. The result row consists of *source*, source of an activity. *Target*, target of an activity. *Frequency*, occurrences of each sequence of processes. And *dependency*, dependecy value between sequence of processes. Since, the dictionary consists of key and value, key is

"actA → actB" and value is total occurrences of each sequence of processes. Thus, the key will be split into *rowResult.source* (step.27) and *rowResult.target* (step.27). Lastly, frequency will be populated with value/total occurrences of each sequence of processes.

Algorithm 2 Business Process Discovery

```
1:  procedure BusinessProcessDiscovery(logPath)
2:      logData = ReadLogFile(logPath)
3:
4:      // get sequence of processes and occurrences
5:      while this is not the end of logData do
6:          row = current row of the logData
7:          id = null
8:          if id != row.id then
9:              // Get a TraceId for each clusters
10:             id = row.id
11:             actA = row.Activity
12:             actB = null
13:         else
14:             actB = row.Activity
15:             key = actA + '->' + actB
16:             if dict.exists(key) then dict[key] += 1
17:             else dict.add(key, 1)
18:             id = row.id
19:             actA = actB
20:         go to the next row
21:
22:     // create frequency table
23:     resultTable = create new result table
24:     while this is not the end of the dictionary do
25:         dictItem = current dictionary entry
26:         rowResult = new result table row
27:         rowResult.source = dictItem.key.split(' -> ')[0]
28:         rowResult.target = dictItem.key.split(' -> ')[1]
29:         rowResult.frequency = dictItem.value
30:         add rowResult into resultTable
31:         go to the next dictionary entry
32:
33:     // calculate dependency
34:     while this is not the end of result table do
35:         rowResult = current row
36:         invertRow = Get rows from result table where row.source = rowResult.target and row.target
    = row.source
37:         dependency = (rowResult.frequency - invertRow.frequency) / (rowResult.frequency - in-
    vertRow.frequency + 1)
38:         rowResult.dependecy = dependecy
39:         go to the next row
40:
41:     return resultTable
```

In the last block of code, the algorithm will calculate the dependency between sequence of processes. Using the formulation that described in definition.11 $a \Rightarrow_L b = (\frac{|a>_L b|-|b>_L a|}{|a>_L b|+|b>_L a|+1})$, we can calculate the dependency between two sequence of processes. First of all, the algorithm will loop the result table and put the current row into *rowResult*. Next, the algorithm will get the invert sequence in the *invertRow*, it means if the current sequence is $a >_L b$ then the

invertRow will be $b >_L a$. In step.37 we can get the dependency factor between sequence of processes using the formulation that we mentioned before. Lastly, we update the result table with the new dependency factor that we got before. In the end, we can successfully generate the result table based on the definition.16.

5.1.2 Workflow Related Elements Discovery Algorithm

In this algorithms, first of all, we read information of log data (Algorithm.1). And we grouping the data based on *activity* and *login* (Algorithm.3) or *activity* and *application name* (algorithm.ref4) and we count total grouping. The result of grouping will be looped and populate new result row. It will fill *source* as the origin of element with *login/application name* and *target* with *activity*. It also populate *frequency* as total count of grouping and put 0 as default number of *dependency*. We will not use dependency for this algorithm and the reason we keep populating it because we want to minimise creating new data schema and we want to reuse previous data schema, as mentioned in Algorithm.2.

Algorithm 3 Workflow related elements discovery (for Business Actor)

```
procedure WorkflowRelatedElements(logPath)
    logData = ReadLogFile(logPath)
    var tmpResult = group by Activity, login and select Activity, login, count
    while this is not the end of tmpResult do
        rowTmp = current row
        newResultRow.source = rowTmp.login
        newResultRow.target = rowTmp.Activity
        newResultRow.freq = count
        newResultRow.depResult = 0
        insert newResultRow to resultTable
        go to the next row

    return resultTable
```

Algorithm 4 Workflow related elements discovery (for Application Component)

```
procedure WorkflowRelatedElements(logPath)
    logData = ReadLogFile(logPath)
    var tmpResult = group by Activity, AppName and select Activity, AppName, count
    while this is not the end of tmpResult do
        rowTmp = current row
        newResultRow.source = rowTmp.AppName
        newResultRow.target = rowTmp.Activity
        newResultRow.freq = count
        newResultRow.depResult = 0
        insert newResultRow to resultTable
        go to the next row

    return resultTable
```

5.1.3 EA Analysis Functions Algorithm

```
public static void CalculateResponseTime(String logPath, ref ElementTable eTable)
{
    List<SimpleData> datas = LogService.GetDataList(logPath);
    var calcTable = datas.GroupBy(g => g.Activity)
        .Select(s => new { Activity = s.Key, calc = String.Format("{0}\r\nAvgR={1},
            MinR={2}, MaxR={3}, SumR={4}", s.Key, s.Min(p => p.ResponseTime),
            s.Min(p => p.ResponseTime), s.Max(p => p.ResponseTime), s.Sum(p => p.ResponseTime)) });
    eTable.ElementRows.ForEach(eT => eT.ElementAlias =
```

```
        calcTable.FirstOrDefault(cT => cT.Activity == eT.Element).calc);
}
```

First of all, this algorithm need logPath to get the log and the data from the log is inputted to list of SimpleData (Algorithm.1). From the list of SimpleData class the algorithm will calculate summation, mean, minimum, and maximum of *ResponseTime* categorised based on *activity*. Next, the analysis function result will be presented into a string and will be updated into the element table, field ElementAlias. The alias will be used in the Archi conversion to add information of the element.

5.2. Archi File Generation

In the following sections we will discuss about the implementation of Archi file generation, as the name suggested the objective of these sections are to produce an Archi file. Archi[1] is open-source modelling tools for Archimate. We first discuss how Archi is structured, and then we talked about the generation process.

5.2.1 Archi Metamodel

In order to create an automated conversion, we used the the low level structured of the Archi file. It is represent as an xml file. Thus, in creating the EA model using Archi is same as generating an xml file with bounded to rules and structured that is defined by Archi. Hence, what is the metamodel of Archi? Archi structured as follows, it has the root element of *model* and it consists of attributes of *name*, the name of the model. *id*, unique identification in global unique identifier (GUID), *version*, the archi version, and lastly sequence of elements of *folder*. Each folders represents a layer in Archimate that denotes in *type*, and *name* attribute. it also has *id*, in GUID. Each *folders* consists of zero to many *elements*. It has attributes of *name*, name of the element, id, and *source* and *target*. Source and target is used to describe Archimate's relationships. From which element (source) to what element (target). Each elements also may or may not include *child*, usually a *child* is used to define view in the Archi. A *child* consists of id, *targetConnections*, *archimateElement*. A *targetConnections* is to create connectivity between *sourceConnection* to the child, while an *archimateElement* represents which Archimate Element that the child is. A child also has *bounds*, a *bound* represent location of the child in *x* and *y*. The *width* and the *height* of the child. Lastly the child has *sourceConnections*, that represent connectivity between the child to other children. The sourceConnection consists of id, *source*, *target*, *archimateRelationship*. A *target* depicts connections from other children to the child, and *target* from the child to other children. Lastly, which *archimateRelationship* that used to describe relationship between the child to other children. The metamodel can be seen at Fig.5.3.

```
<?xml version="1.0" encoding="UTF-8"?>
<archimate:model xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:archimate="http://www.archimatetool.com/archimate" name="(new model)" id="2ac9bad3-29b4-4eab-9259-87e17db526cf" version=
"4.0.0">
  <folder name="Strategy" id="d29a1261-b5c2-4f78-8e18-79b356250330" type="strategy"/>
  <folder name="Business" id="f4f9b6fd-dcf4-4244-88d6-89c943b04308" type="business">
    <element xsi:type="archimate:BusinessActor" name="Business Actor" id="5f4e8f53-6a22-4042-b897-35c228c04385"/>
    <element xsi:type="archimate:BusinessProcess" name="Business Process" id="56150bfa-f585-4b18-bde1-204b9a88e865"/>
  </folder>
  <folder name="Application" id="c402081c-3120-4891-82ce-4a84644d640e" type="application"/>
  <folder name="Technology &amp; Physical" id="cb00180d-fb16-4d71-8f1e-54fb9706d592" type="technology"/>
  <folder name="Motivation" id="c07c325e-228a-40e1-977b-64943d5e25f3" type="motivation"/>
  <folder name="Implementation &amp; Migration" id="d004929e-896f-4d16-8dec-1de95051af03" type="implementation_migration"/>
  <folder name="Other" id="472e6d18-7525-4073-8354-897ac9087fa5" type="other"/>
  <folder name="Relations" id="ba6929dd-dc46-47ca-8a65-80fce96e8604" type="relations">
    <element xsi:type="archimate:AssignmentRelationship" id="3c3bf568-71c3-4a21-b85d-bd79e8eed90b" source="5f4e8f53-6a22-4042-b897-35c228c04385" target="56150bfa-f585-4b18-bde1-204b9a88e865"/>
  </folder>
  <folder name="Views" id="68b748ee-d345-46d8-8ebc-3c9973b5e174" type="diagrams">
    <element xsi:type="archimate:ArchimateDiagramModel" name="Default View" id="ce90726c-847c-4b3e-8f0c-75b2e6b303b7">
      <child xsi:type="archimate:DiagramObject" id="79b66bdb-f9b1-4ff0-b0e8-6fad27767b66" archimateElement="5f4e8f53-6a22-4042-b897-35c228c04385">
        <bounds x="364" y="275" width="120" height="55"/>
        <sourceConnection xsi:type="archimate:Connection" id="9951a63c-f5f1-41a3-8c2c-5ab64911fb2e" source="79b66bdb-f9b1-4ff0-b0e8-6fad27767b66" target="53ea6324-3c7d-43d0-b359-762e1d12f346"
        archimateRelationship="3c3bf568-71c3-4a21-b85d-bd79e8eed90b"/>
      </child>
      <child xsi:type="archimate:DiagramObject" id="53ea6324-3c7d-43d0-b359-762e1d12f346" targetConnections="9951a63c-f5f1-41a3-8c2c-5ab64911fb2e" archimateElement="56150bfa-f585-4b18-bde1-204b9a88e865">
        <bounds x="539" y="414" width="120" height="55"/>
      </child>
    </element>
  </folder>
</archimate:model>
```

Figure 5.2: Simple Archi xml

Moreover, in Fig.5.2 is an example of xml that used by Archi. This xml depicts the EA model of two Archimate relationship of *Business Actor* and *Business Process*, with an *Assignment* relationship. As can be seen at the figure, each folder represents Archimate's layer, each

[1]Archi 4.2.0. `https://www.archimatetool.com/`

layer has elements that describe Archimate's elements. The folder also describes the relationships and views that used in the model. In addition, the xml also describes how to define a view and how to positioning elements, size of each element, source and target connectivity between elements.

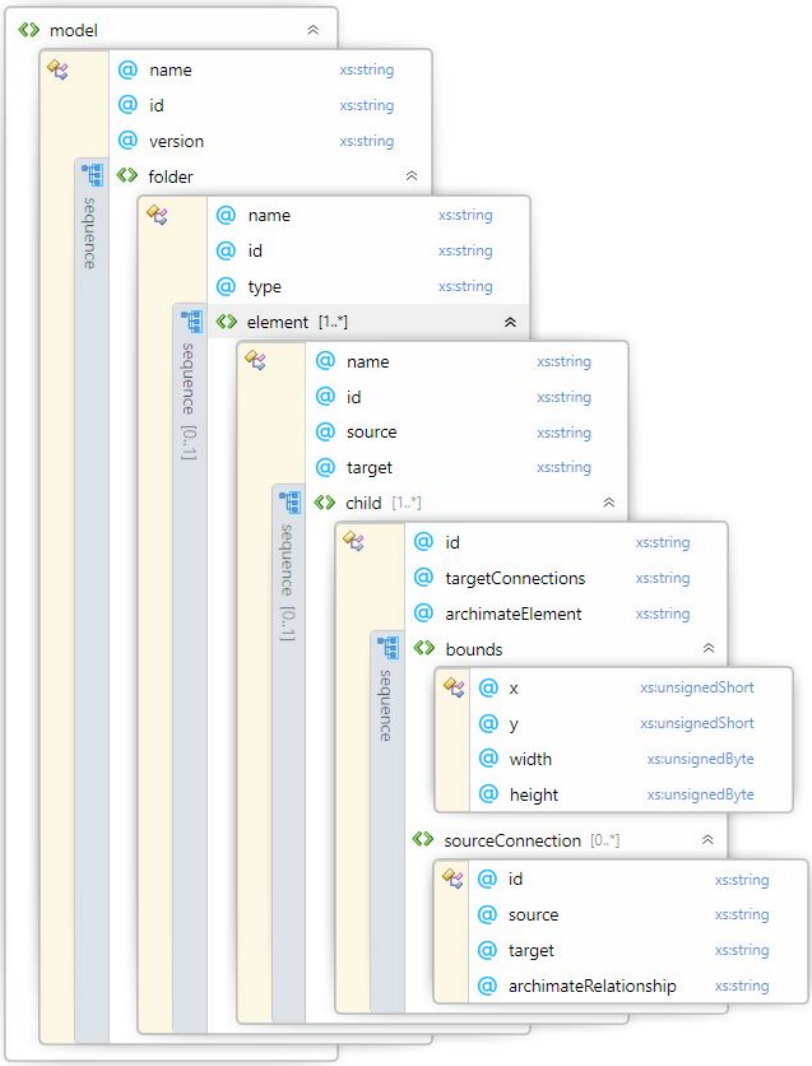


Figure 5.3: Archi metamodel

5.2.2 Generate Relationships and Elements Table

After we understand how Archi works, the next step that necessary to generate automated EA model is to produce a relationship table and elements table. As described previously at the end of the business process discovery algorithm (Algorithm.2) or identification of two elements (Algorithm.3/4) we have result table, which consists of *source, target, frequency*, and *dependency*. Both *source* and *target* in the context of business process discovery is a sequence of business processes. Frequency is total occurrences of a sequence of processes, while dependency defines how dependent relationship between processes.

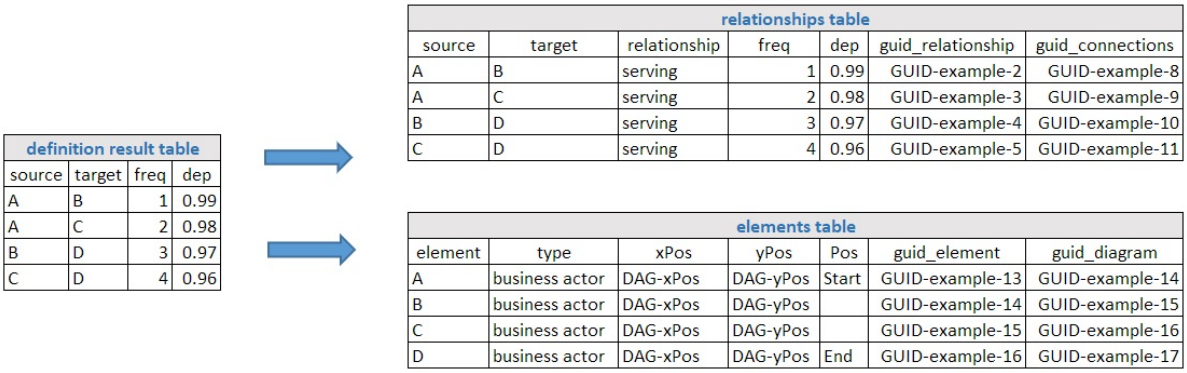
The usage of relationships and elements table is to ease the conversion process, since the Archi dependent on GUID elements to identify its elements, then one of the purpose of this table generation is to incorporate GUID variables before conversion process. In a relationship table we have *source, target, relationship, frequency, dependency, GUID_relationship*, and *GUID_Connection*. As can be seen at Fig.5.4, the relationship table basically extension of result table, with additional attributes. A *Relationship* defines which archimate relationship

of source and target. Both *GUID_relationship*, and *GUID_connection* is to facilitate Archi xml as we discussed previously in Archi Metamodel. While an element table, we have *elements, type, xPos, yPos, position*, *GUID_element*, and *GUID_diagram*. *Elements* is an archimate element. *Type* defines which archimate element that the element is. An *xPos* and *yPos* is to identify location of the element in the view. Position defines the starting and ending position of the elements. Lastly, the GUIDs are used to ease the conversion process.

Algorithm 5 Generate Relationship Table

```
procedure GenerateRelationshipTable(resultTable)
    while this is not the end of result table do
        rowResult = current row
        newRelationshipRow.source = rowResult.source
        newRelationshipRow.target = rowResult.target
        newRelationshipRow.relationship = "Triggering"/"Serving"/"Assignement"
        newRelationshipRow.freq = rowResult.freq
        newRelationshipRow.depResult = rowResult.dep
        newRelationshipRow.guidRelationship = new guid()
        newRelationshipRow.guidConnection = new guid()
        insert newRelationshipRow to relationshipTable
        go to the next row

    return relationshipTable
```



definition result table			
source	target	freq	dep
A	B	1	0.99
A	C	2	0.98
B	D	3	0.97
C	D	4	0.96

relationships table						
source	target	relationship	freq	dep	guid_relationship	guid_connections
A	B	serving	1	0.99	GUID-example-2	GUID-example-8
A	C	serving	2	0.98	GUID-example-3	GUID-example-9
B	D	serving	3	0.97	GUID-example-4	GUID-example-10
C	D	serving	4	0.96	GUID-example-5	GUID-example-11

elements table						
element	type	xPos	yPos	Pos	guid_element	guid_diagram
A	business actor	DAG-xPos	DAG-yPos	Start	GUID-example-13	GUID-example-14
B	business actor	DAG-xPos	DAG-yPos		GUID-example-14	GUID-example-15
C	business actor	DAG-xPos	DAG-yPos		GUID-example-15	GUID-example-16
D	business actor	DAG-xPos	DAG-yPos	End	GUID-example-16	GUID-example-17

Figure 5.4: Fragment of relationships and elements table

In generating relationship algorithm (Algorithm.5), we populated rows from result table into relationship table, the algorithm loop all rows in the result table and create new relationship row, the process simply move all fields into correspoding fields of relationship row. It populates columns: *source, target, frequency*, and *dependency*. The process also put hard-coded value of "Triggering" (for business process discovery), "Serving" (for Workflow related elements discovery (for Application Component)), "Assignment" (for Workflow related elements discovery (for Business Actor)) into the *relationship* field, and instantiate guid value for *guid_relationship* and *guid_connection*. Illustration for this process can be seen at Fig.5.4.

In generate element algorithm, as illustrated in Fig.5.4. We created a dictionary that consists of key and value pair, to accommodate unique list of entries. First of all, the algorithm will loop all rows in the result table and check whether source/target as an element already exists in the dictionary. If it is not exists then source/target will become new entry of the dictionary. The algorithm will create new element row, and populate its field with source/target as element field, hard-code "Business Process" (for business process discovery), "Application Component" (for Workflow related elements discovery (for Application Component)), "Business Actor" (for Workflow related elements discovery (for Business Actor)) to denotes that this element is business process elements. Next, this algorithm will check whether this element

is start/end element based on whether there is any preceding or following elements. After that, the algorithm will populate element table based on all entries from dictionary, it will populate element based on source/target. Type as ”Business Process”. an X-position and y-position is 0, since we put default value 0 as axis coordinate. Position to determine whether the element is starting or ending element, lastly guid value for Archi conversion.

Algorithm 6 Generate Element Table

```
procedure GenerateElementTable(resultTable)
    dictElements = create new dictionary which consists of key and value

    while this is not the end of result table do
        rowResult = current row
        if dictElements.exists(rowResult.source) then
            row = new elementRow()
            row.Element = rowResult.source
            row.Type = ”Business Process”/”Business Actor”/”Application Component”
            if count(resultTable where resultTable.target = rowResult.source) then
                row.Position = ”Start”
            dictElements.add(rowResult.source, row)

        if dictElements.exists(rowResult.target) then
            row = new elementRow()
            row.Element = rowResult.target
            row.Type = ”Business Process”/”Business Actor”/”Application Component”
            if count(resultTable where resultTable.source = rowResult.source) then
                row.Position = ”End”
            dictElements.add(rowResult.source, row)
        go to the next row

    while this is not the end of dictElements do
        entryDict = current dictionary entry
        newElementRow.Element = entryDict.value.Element
        newElementRow.Type = entryDict.value.Type
        newElementRow.xPos = 0
        newElementRow.yPos = 0
        newElementRow.Position = entryDict.value.Position
        newElementRow.guidElement = new guid()
        newElementRow.guidDiagram = new guid()
        insert newElementRow to elementTable
        go to the next row

    return elementTable
```

5.2.3 Archi File Generation

We generated an Archi file based on result of (Algorithm.2,3,4) and the result will be converted into relationship table (Algorithm.5) and element table (Algorithm.6). The Archi will take *exportPath* or location for export, *relationship table* and *element table* as inputs. The algorithm will create an xml file based on xsd schema (Fig.5.3). First of all, it will generate xml element of folder with element name as *business* to denotes business layer construct, and all elements from element table that corresponding to this layer will be populated. The same procedure also exists for application layer. Next, the algorithm will produce xml element of folder with element name as *relations* to cluster all relationships of relationship constructs. This algorithm will populate all information from the *relationship table*. Lastly, this algorithm will produce xml element of *diagram*, to denotes *viewpoint* that illustrate elements,

relationship of Archimate. Algorithm for Archi conversion can be seen at Appendix.C.

5.3. Prototype

In the treatment design phase, we develop all the algorithms, code implementations, and Archi file generation using Microsoft C# programming language. We develop a desktop based application with .NET Framework 4.5.2. The application has four user interfaces. First, the main user interface (Fig.5.5), it has three features: log files selection, export locations selection, algorithms selections. Next, the application has three separate tabs, *result, relationship*, and *element* tabs. *Result* (Fig.5.6) is used to display the result of business process discovery algorithm (algorithm.2). Relationship (Fig.5.7) and Element tabs (Fig.5.8) are used to display result of generate relationship and element algorithm (algorithm.5 and 6).

The prototype processes are executed sequentially. First, it will executed business process discovery algorithm (BPD) to create a result table with an event log as an input, and it will call generate relationships table with input of the result table and produces a relationship table. Next, it will execute the generate elements table with the relationship table as an input and it produces element table as an output. After that, the element table will be calculated using analysis function. After BPD is done, the prototype then execute workflow related elements discovery (WRED) and it also used the same event log and produces a result table, next it also execute generate relationships and elements table to produce relationship and element table. After the WRED is finished, the prototype then generate the Archi file. First of all, the prototype will combine all tables from BPD and WRED and produced the combined version of result, relationship, and element tables. Then all the tables will be converted into an EA model. The illustration of this process can be seen at Fig.5.9.

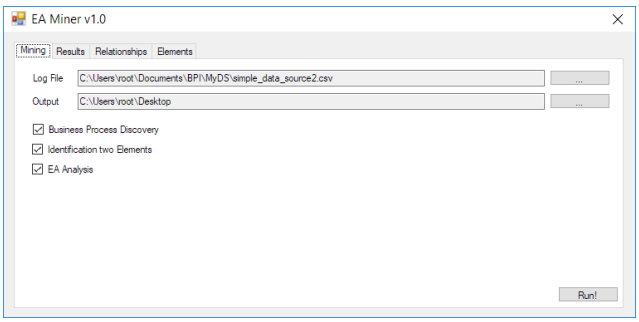


Figure 5.5: Main user interface

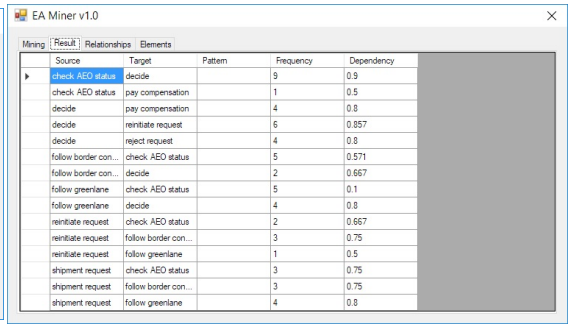


Figure 5.6: Result table interface

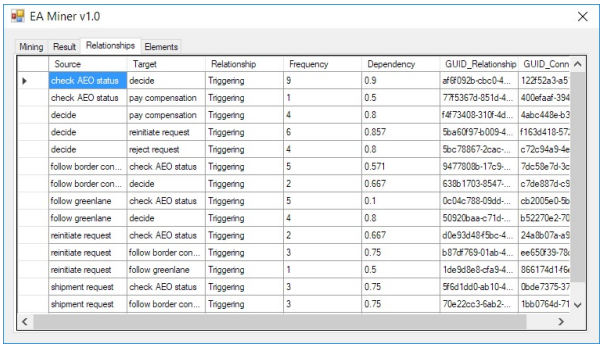


Figure 5.7: Relationship table interface

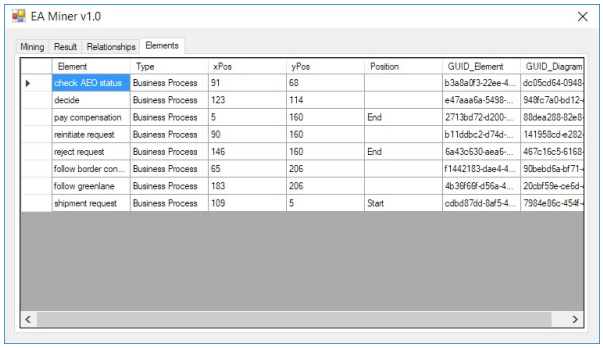


Figure 5.8: Element table interface

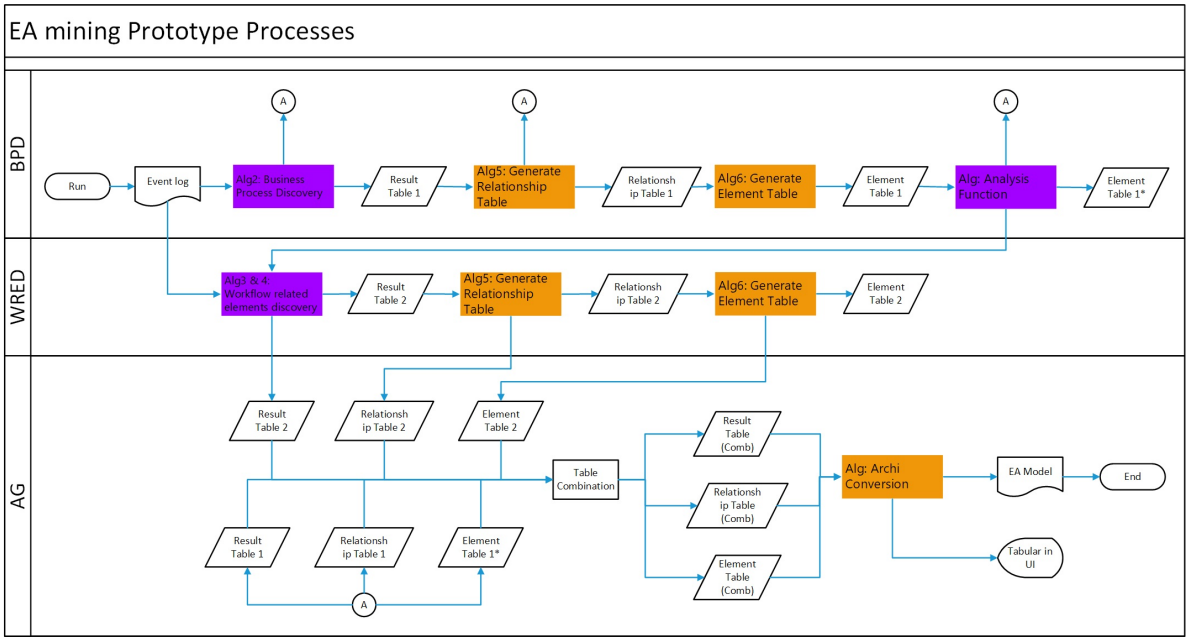


Figure 5.9: Prototype processes

Acronyms on the swimlane are: Business process discovery (BPD), Workflow related elements discovery(WRED), Archi generation (AG)

6

Validation

This chapter will talk about the validation process that are conducted during the research, this validation used single-case mechanism to test EA mining. The validation was also conducted comparison pattern to test the conversion pattern that were used to associate a process model to an EA model. Lastly, The validation conducted performance test to test the prototype.

6.1. EA mining validation

Figure 6.1: EA mining validation workflow

In order to validate the research, we used a single-case mechanism experiment. The experiment is a test of a single case in which the researcher applies stimuli to the case and explains the responses in terms of mechanisms internal to the case [45]. The steps to test the log structure are: first, we tried to use the guideline (Section.4.1.4) to produce an event log based on a case study. After that we used EA mining to produced an EA model. Finally, the result of the EA model then will be compared with the perceived business process, infrastructure, and stakeholder. Steps for this workflow can be seen at Fig.6.1. The case that we used is MyShop, an imaginary e-commerce company that runs e-Shop in B2C environment. The company has business process that consists of *browsing product and placing orders*, *order payment*, *order shipping*, and *record sales*. Each activity has information regarding the order, for example in order placing there are the order number, along with products code, order date, transaction amount, receiver name, shipment address. There are also information regarding the process, for example user who access this activity, application name of the activity, process date time that the process is executed, and ResponseTime is that needed for the system executed the process. The business process can be seen at Fig.6.2 along with information that the process has.

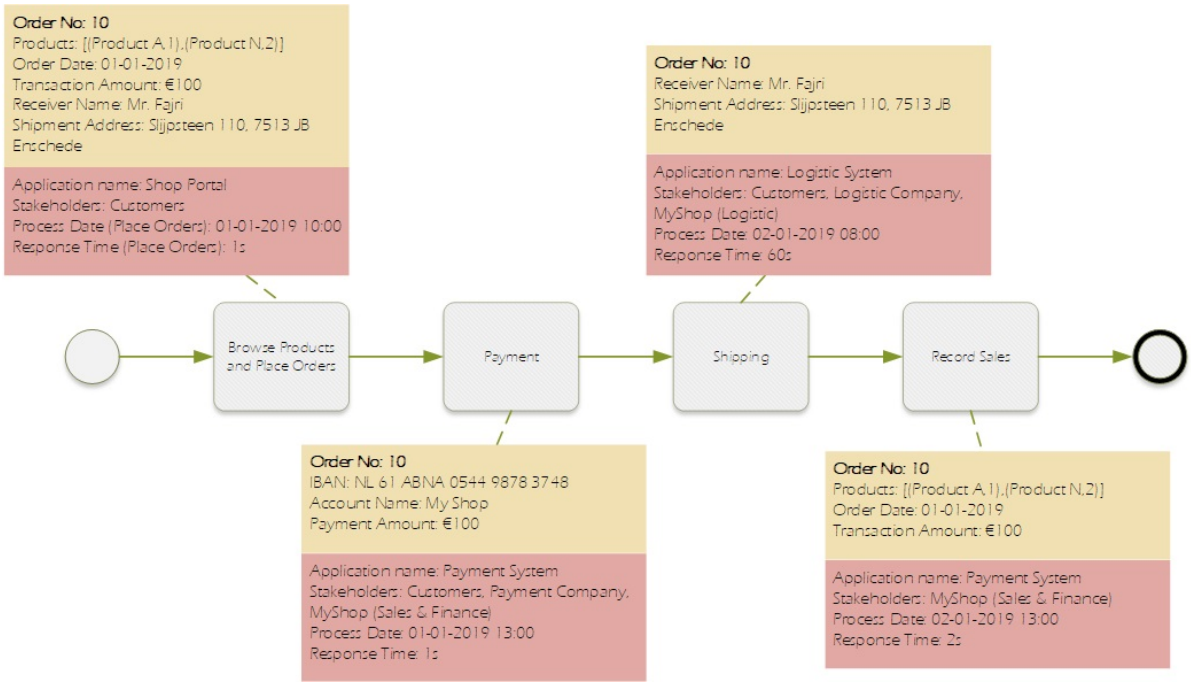


Figure 6.2: Simple e-commerce transactions

6.1.1 Step 1: Log generation

There are four steps in the guideline, *identify a business model, identify activities, key, and workflow, identify trace and events, collecting relevant attributes and create a log* (Figure.4.3). Each steps of the guideline is used to generate a log that can be used for EA mining conversion. Each steps is explained in detail in the following sections.

6.1.1.1 Step 1a: Identify a business model

The business model is e-commerce B2C e-shop. It consists of key activities of *browse products and place orders, payment, shipping,* and *record sales.*

6.1.1.2 Step 1b: Identify activities, key, and workflow

In this step there are three sub-steps: to identify activities, to identify key that link those activities, and to identify workflow, how those activities interact with each other. In identification of activities, the step that needed is to look for key activities that the business model have. In the case of MyShop there are four activities: *browse products and place orders, payment, shipping,* and *record sales.* These activities also executed in sequential order, that phenomenon can be called a workflow. In some case, there might be many activities and combination of those activities can produce different workflows (see Fig.6.4). For the purpose of this research, MyShop only have one workflow with activities that are executed in sequence. The workflow is sequential order from browse products to record sales. Lastly, each activities have a key to link between those activities in a workflow, in MyShop the key to link between activities is an *order number.* Illustration of this process can be seen at Figure.6.3, the red boxes show activities, the green box shows workflow, and the blue boxes show keys.

6.1.1.3 Step 1c: Identify traces and events

Traces are workflows that available within observation, in MyShop there are only one *Trace* with Order Number as an identifier, and *Events* will represented as each activities that the trace has, each events can have an arbitrary number to identify itself. In Fig.6.3, the green box is a workflow with key (the blue boxes) as its identifies. This workflow is identified as a

trace in the log structure (Table.6.1), and each events is shown as the red boxes. Since the case did not specified the identification number for each events, thus this case is used an arbitrary number to identify each events (see Table.6.1).

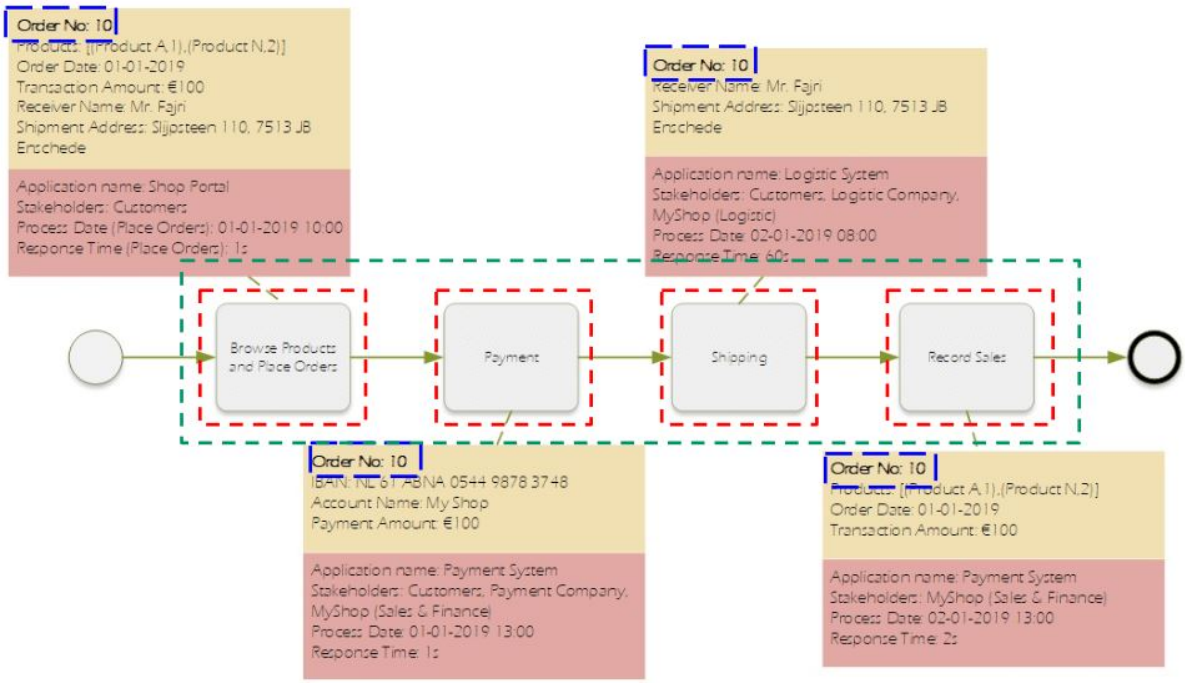


Figure 6.3: Identify activities, key, and workflow

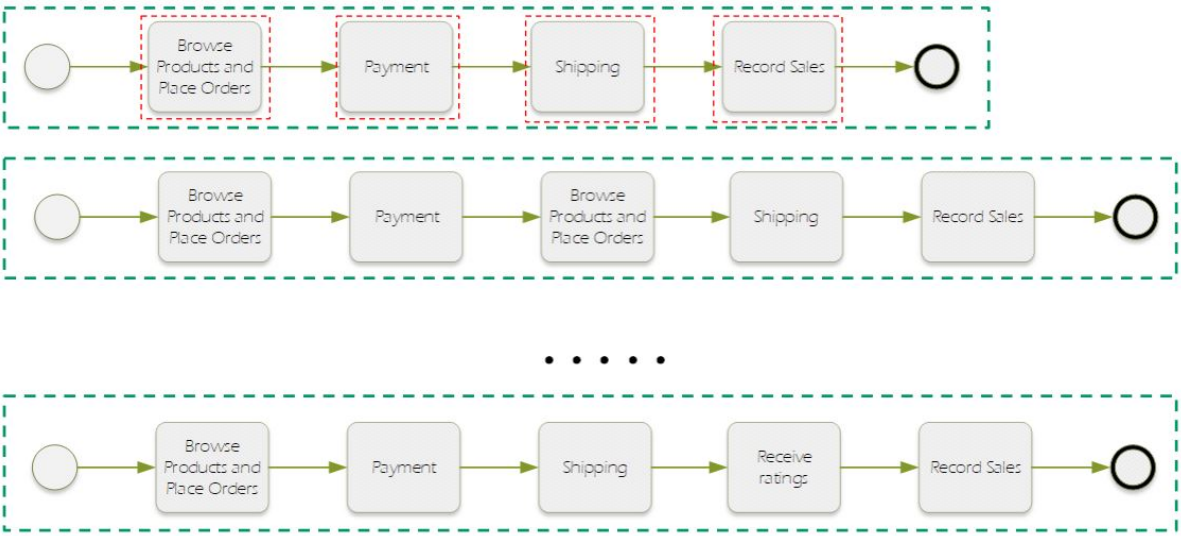


Figure 6.4: Example of multiple traces and events

6.1.1.4 Step 1d: Collecting relevant attributes and create a log

Next process that need to be done is collecting relevant attributes. Since this research already have a structure to define event log (Section.4.1.1), then to create an event log, this guideline is helping to fill the data into event log that follows the log structure. The structure have TraceId which are filled by key (Order number). EventId is filled with an arbitrary number, since MyShop did not specify identifier for each event. Next, each events has attributes

Activity, Timestamp, Login, AppName, and *ResponseTime*. Each attributes can be populated from MyShop data. Activity is the activity name, Timestamp is the process date, Login is Stakeholder, AppName is application name, and response time is response time of the application to execute each process. The blue boxes from Fig.6.5 is data that needed to fill an event's attributes. Thus, after filling all the data an event log can be created (Table.6.1).

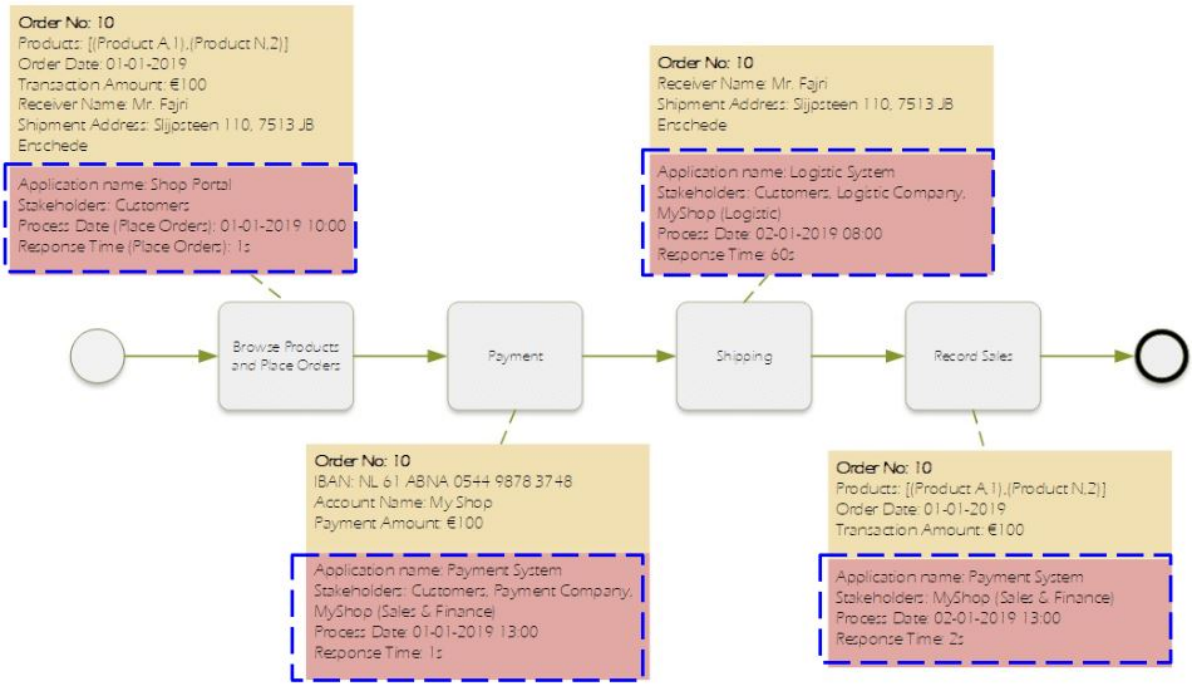


Figure 6.5: Collecting relevant attributes

Table 6.1: MyShop event log

TraceID	EventID	Activity	Timestamp	Login	AppName	Response Time
10	1	Browse Products and Place Orders	01-01-19 10:00	Customer	Shop Portal	1
10	2	Payment	01-01-19 13:00	Customer	Payment System	1
10	3	Shipping	02-01-19 08:00	MyShop(Logistic)	Logistic System	60
10	4	Record Sales	02-01-19 13:00	MyShop (Sales and Finance)	S/4 HANA Finance	30

6.1.2 Step 2: EA model creation

We created an EA model from the log (Table.6.1). We used the prototype that implement both definition and algorithm of EA mining. The EA model can be seen at Fig.6.6. As can be seen at the figure, there are four business processes, four application components, and three business actors. The EA mining successfully generate results of the three steps of EA mining, *business process discovery* that discovery sequence of processes. *Workflow related elements discovery* that able to discover elements that related to the workflow, and lastly, *EA analysis function* to conduct simple analysis of EA elements.

6.1.3 Step 3: Model comparison

In the last step, we compared business process that we perceived in the MyShop and we also list down necessary information that related to the business processes. The business process and the information can be seen at Fig.6.2. The perceived business process is same with the EA model that created based on log (Fig.6.6). Moreover, additional information that

we gathered from the enterprise that were inputted into the log can also be extracted into relevant EA elements.

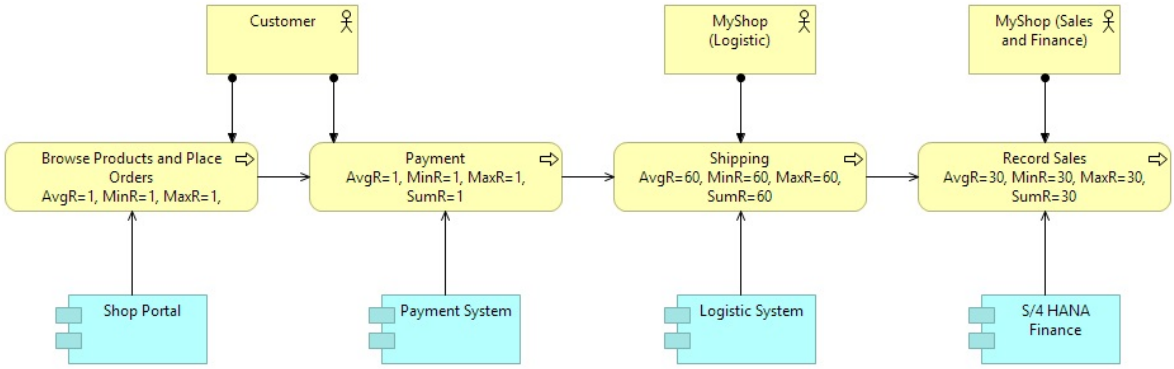


Figure 6.6: MyShop EA model

6.2. Conversion pattern validation

In the literature study, we propose conversion pattern of the dependency graph into EA elements (Section.2.2.4), this validation is used to validate the accuracy of EA mining if it is converted from a dependency graph.

6.2.1 Validation #1 - Small dataset

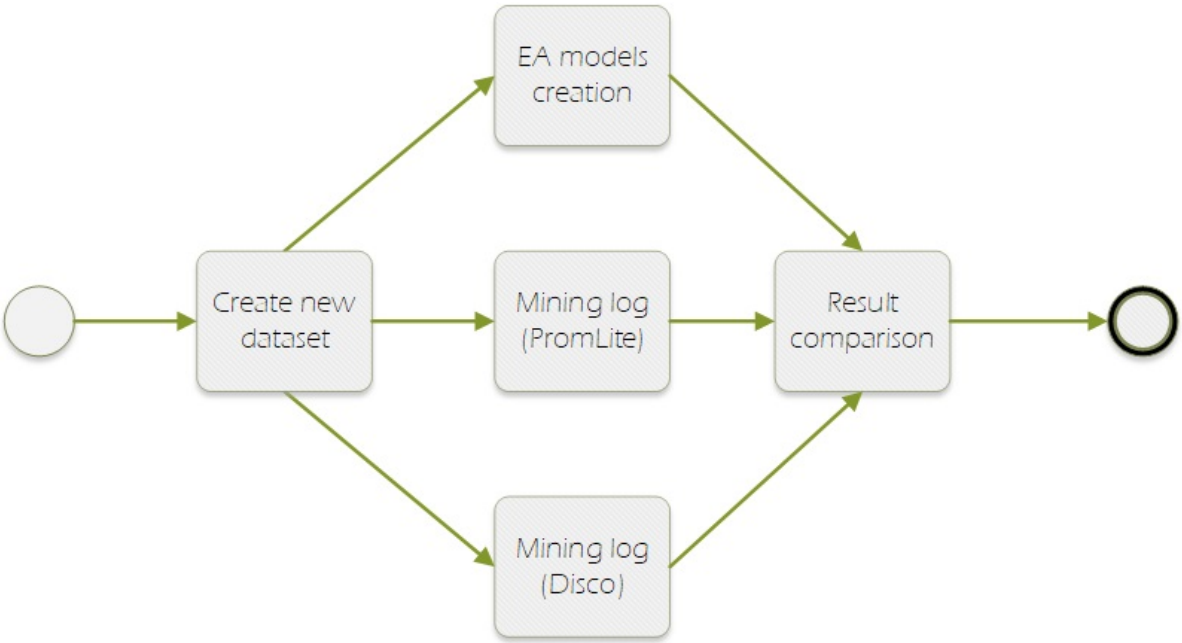


Figure 6.7: Conversion pattern test #1 workflow

In this validation, we used an arbitrary dataset that consists of 71 events, 10 traces, and eight Activities. The objective of this validation was to evaluate the validity of our EA model results comparing to other process mining applications. The testing processes were: we created a dataset based on the log structure 4.2, which consists of *TraceId, Activity, Resource, Timestamp, AppService*, and *AppName*. We populated the dataset with arbitrary values from logistic processes. After that, we used our application to process the log. We also created process models using disco and promlite, then we compare the results (Fig.6.7). Since there

were no available tools for the automated comparison, then we have to do the validation manually. From the result table that was produced from our app, we check if the sequence of processes with the same frequency and dependency also occur in other results.

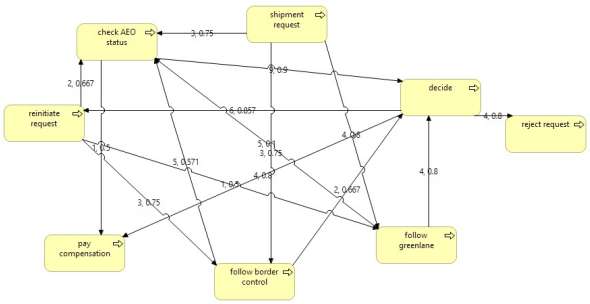
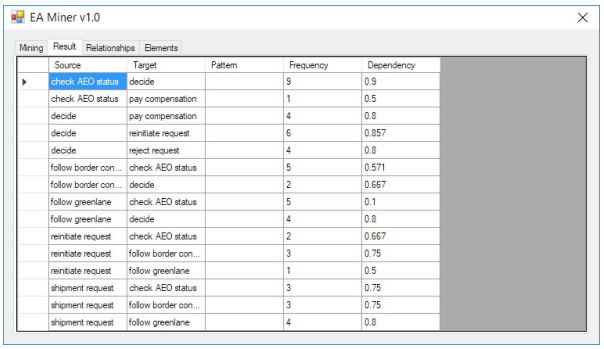


Figure 6.8: EA miner result



EA Miner v1.0

Mining | Result | Relationships | Elements

Source	Target	Pattern	Frequency	Dependency
check AEO status	decide		9	0.9
check AEO status	pay compensation		1	0.5
decide	pay compensation		4	0.8
decide	reinitiate request		6	0.857
decide	reject request		4	0.8
follow border con...	check AEO status		5	0.571
follow border con...	decide		2	0.667
follow greenlane	check AEO status		5	0.1
follow greenlane	decide		4	0.8
reinitiate request	check AEO status		2	0.667
reinitiate request	follow border con...		3	0.75
reinitiate request	follow greenlane		1	0.5
shipment request	check AEO status		3	0.75
shipment request	follow border con...		3	0.75
shipment request	follow greenlane		4	0.8

Figure 6.9: Result table of business process discovery

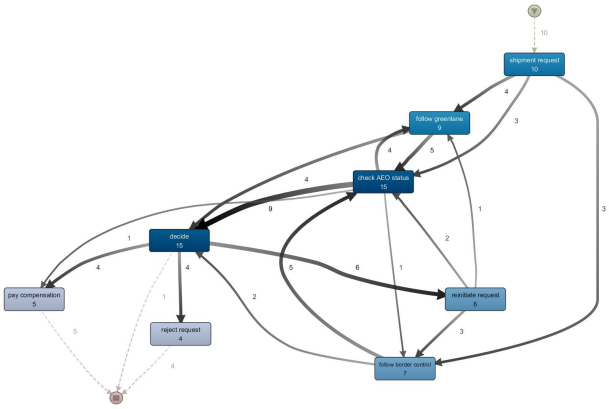


Figure 6.10: Fluxicon-Disco Result

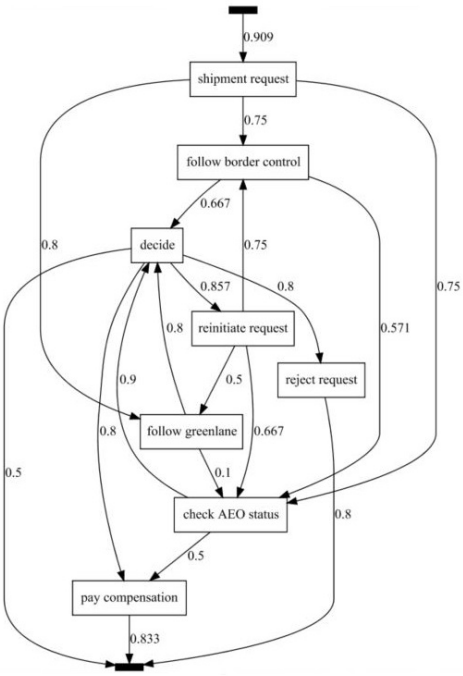


Figure 6.11: PromLite Interactive Heuristic Result

Figure.6.9 depicted the result of business process discovery algorithm (algorithm.2). And Archi conversion (Appendix.C) was shown by Fig.6.8. The result also same if compared to disco (Fig.6.10) and promlite (Fig.6.11). Each sequence of processes was same between our EA model and other applications' models. In promlite (Fig.6.11) the dependency graph showed the dependency factor, while disco shown frequency of activities, which was also the same with our finding.

6.2.2 Validation #2 - Larger dataset

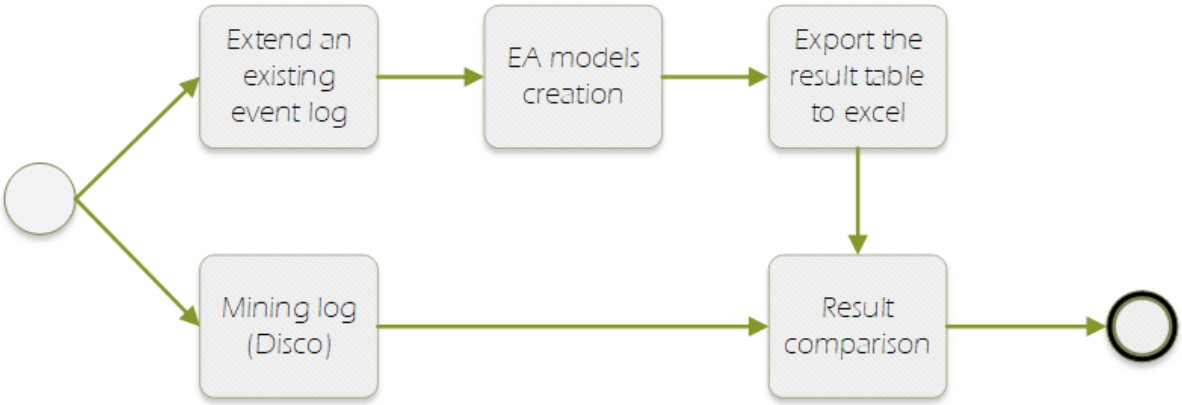


Figure 6.12: Conversion pattern test #2 workflow

In this validation we used larger dataset [1]. The dataset consists of 262.000 events, 13.087 cases, 36 activities, 5 users, and 5 applications. In this test we extend the dataset, the dataset already contain TraceId, EventId, Activity, Timestamp and Login. Hence, we add additional attributes (AppName, ResponseTime) to accommodate the EA mining conversion. The flow of this test are: First, we extend the existing dataset. Next, we create EA models. Since the result of algorithms is quite complex to check it manually (Fig.6.13), then we only highlight three activities for our validation. We used disco as our comparison tool and we extract relationship result from application and put it in excel to ease the comparison process, and we compare incoming and outgoing relationship with its frequency, the testing workflow can be seen at Fig.6.12. The results of this test are: all incoming and outgoing is same both of our application and disco. For example, for "*A _PREACCEPTED_COMPLETE*" there are two incoming and one outgoing relationship (Fig.6.14), the result is same as our application (Fig.6.14), there are one "*A _PREACCEPTED_COMPLETE*" in source for "Triggering" relationship and there are two in target, also the frequency produces the same result as in disco. The comparison produces same result for "*O_CREATED_COMPLETE*" (Fig.6.16, 6.16) and "*W_Neballen Offertes-SCHEDULE*" (Fig.6.18, 6.18).

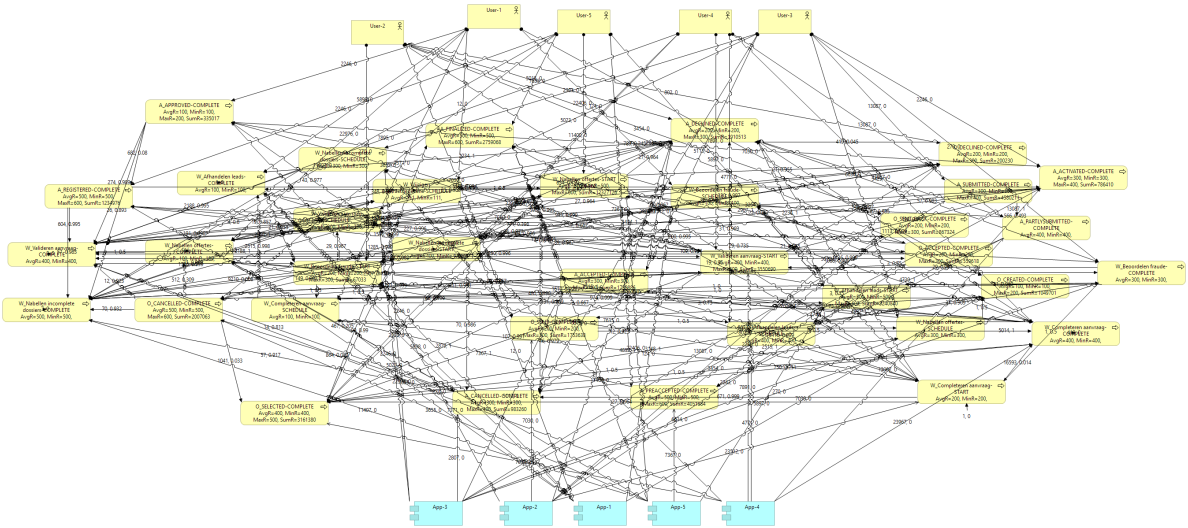


Figure 6.13: BPI 2012 result

[1] https://www.win.tue.nl/bpi/doku.php?id=2012:challenge

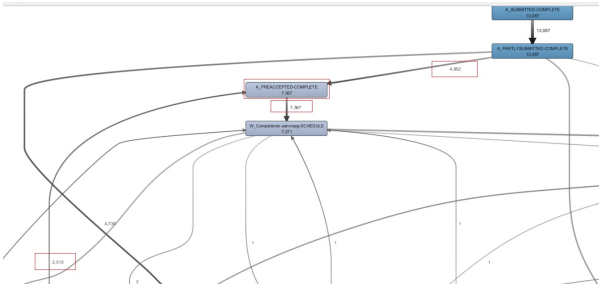


Figure 6.14: BPI 2012 validation #1 (disco)

key	source	target	relationsl	freq	deg
A_PARTLYSUBMITTED-COMPLETE A_PREACCEPTED-COMPLETE	A_PARTLYSUBMITTED-COMPLETE	A_PREACCEPTED-COMPLETE	Triggering	4852	1
A_PREACCEPTED-COMPLETE W_Completeren aanvraag-SCHEDULE	A_PREACCEPTED-COMPLETE	W_Completeren aanvraag-SCHEDULE	Triggering	7367	1
W_Afhandelen leads-START A_PREACCEPTED-COMPLETE	W_Afhandelen leads-START	A_PREACCEPTED-COMPLETE	Triggering	2515	1
User-5 A_PREACCEPTED-COMPLETE	User-5	A_PREACCEPTED-COMPLETE	Assignment	7367	0
App-5 A_PREACCEPTED-COMPLETE	App-5	A_PREACCEPTED-COMPLETE	Serving	7367	0

Figure 6.15: BPI 2012 validation #1 (excel)

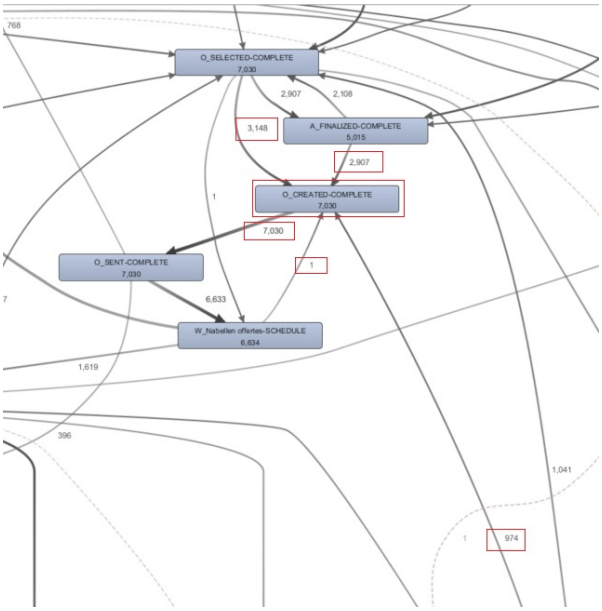


Figure 6.16: BPI 2012 validation #2 (disco)

key	source	target	relationsl	freq	deg
A_FINALIZED-COMPLETE O_CREATED-COMPLETE	A_FINALIZED-COMPLETE	O_CREATED-COMPLETE	Triggering	2907	1
O_CANCELLED-COMPLETE O_CREATED-COMPLETE	O_CANCELLED-COMPLETE	O_CREATED-COMPLETE	Triggering	974	1
O_CREATED-COMPLETE O_SENT-COMPLETE	O_CREATED-COMPLETE	O_SENT-COMPLETE	Triggering	7030	1
O_SELECTED-COMPLETE O_CREATED-COMPLETE	O_SELECTED-COMPLETE	O_CREATED-COMPLETE	Triggering	3148	1
W_Nabellen offertes-SCHEDULE O_CREATED-COMPLETE	W_Nabellen offertes-SCHEDULE	O_CREATED-COMPLETE	Triggering	1	0,5
User-1 O_CREATED-COMPLETE	User-1	O_CREATED-COMPLETE	Assignment	7030	0
App-1 O_CREATED-COMPLETE	App-1	O_CREATED-COMPLETE	Serving	7030	0

Figure 6.17: BPI 2012 validation #2 (excel)

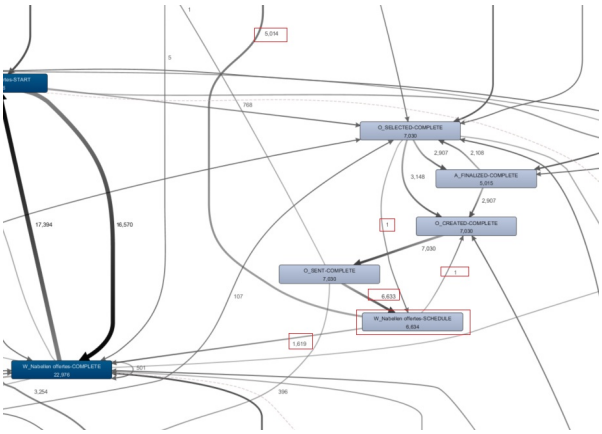


Figure 6.18: BPI 2012 validation #3 (disco)

key	source	target	relationsl	freq	deg
O_SELECTED-COMPLETE W_Nabellen offertes-SCHEDULE	O_SELECTED-COMPLETE	W_Nabellen offertes-SCHEDULE	Triggering	1	0,5 ee
O_SENT-COMPLETE W_Nabellen offertes-SCHEDULE	O_SENT-COMPLETE	W_Nabellen offertes-SCHEDULE	Triggering	6633	1 ac
W_Nabellen offertes-SCHEDULE O_CREATED-COMPLETE	W_Nabellen offertes-SCHEDULE	O_CREATED-COMPLETE	Triggering	1	0,5 7a
W_Nabellen offertes-SCHEDULE W_Completeren aanvraag-COMPLE	W_Nabellen offertes-SCHEDULE	W_Completeren aanvraag-COMPLETE	Triggering	5014	1 2(
W_Nabellen offertes-SCHEDULE W_Nabellen offertes-COMPLETE	W_Nabellen offertes-SCHEDULE	W_Nabellen offertes-COMPLETE	Triggering	1619	1 9f
User-3 W_Nabellen offertes-SCHEDULE	User-3	W_Nabellen offertes-SCHEDULE	Assignment	6634	0 ba
App-3 W_Nabellen offertes-SCHEDULE	App-3	W_Nabellen offertes-SCHEDULE	Serving	6634	0 ac

Figure 6.19: BPI 2012 validation #3 (excel)

6.3. Prototype test

In the next test, we conducted performance validation. The objective of this validation is to test the performance of the prototype. We used two real-life dataset from BPI challenges. We also converted the original dataset to csv format and add additional columns to comply with our algorithms. We put the number of events, cases, activities, users and applications in

Table.6.2. Then we execute the implementation and we calculate the time needed for each algorithms to be executed. The time needed were recorded in unit of seconds, and the result can be seen at Table.6.2. As can be seen at the table, the number of users and application of log is not significantly increase the time needed for the execution, however the number of activities increase the time needed for executions.

Table 6.2: Performance Test

	BPI 2011	**BPI 2011-double users & app**	**BPI 2012**
Events	150.291	150.291	262.200
Cases	1.143	1.143	13.087
Activities	624	624	36
Users	20	40	5
Applications	10	20	5
Alg1-Total	23.29	27.31	10.56
Alg3-Total	64.49	63.66	15.48
Alg2-Total(Actor)	24.33	24.14	9.81
Alg2-Total(App)	23.46	23.78	9.13
ArchiExport	1.83	1.52	0.17
Algorithms Total Time (seconds)	137.41	140.42	45.17

In addition, we wanted to determine the linear relationship between total time that was took in order to execute a dataset. If we convert the table into scatter plot, it can be seen that time variable with other variables have linear relationship (Fig.6.20). Then, we tested the performance test result with bi-variate correlation test. The result of the test can be seen at Fig.6.21. As can be seen from the figure, the algorithm times are positively correlate with number of activities, users, and applications. In contrast, the algorithm times are negatively correlate with number of events and cases.

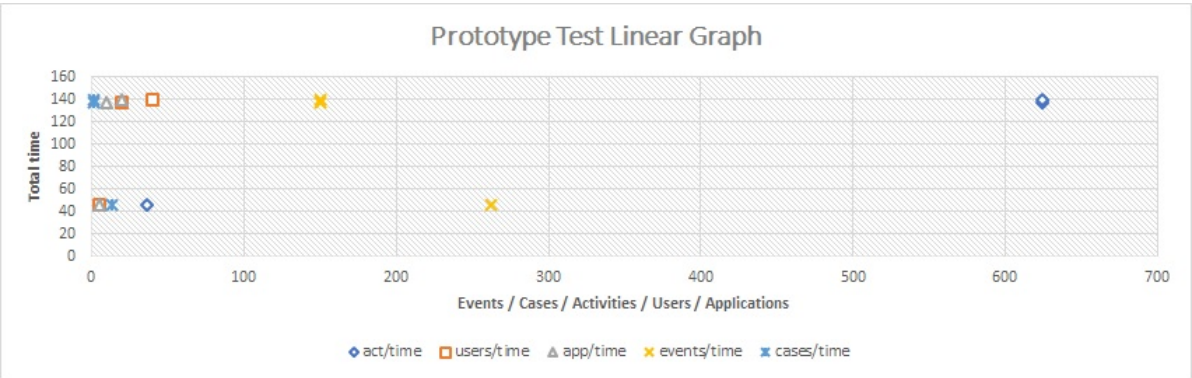


Figure 6.20: Prototype linear graph

Correlations

		Events	Cases	Activities	Users	Applications	Algorithms Total Time (seconds)
Events	Pearson Correlation	1	1.000**	-1.000**	-.822	-.756	-1.000*
	Sig. (2-tailed)		.000	.000	.386	.454	.018
	N	3	3	3	3	3	3
Cases	Pearson Correlation	1.000**	1	-1.000**	-.822	-.756	-1.000*
	Sig. (2-tailed)	.000		.000	.386	.454	.018
	N	3	3	3	3	3	3
Activities	Pearson Correlation	-1.000**	-1.000**	1	.822	.756	1.000*
	Sig. (2-tailed)	.000	.000		.386	.454	.018
	N	3	3	3	3	3	3
Users	Pearson Correlation	-.822	-.822	.822	1	.994	.838
	Sig. (2-tailed)	.386	.386	.386		.069	.368
	N	3	3	3	3	3	3
Applications	Pearson Correlation	-.756	-.756	.756	.994	1	.774
	Sig. (2-tailed)	.454	.454	.454	.069		.437
	N	3	3	3	3	3	3
Algorithms Total Time (seconds)	Pearson Correlation	-1.000*	-1.000*	1.000*	.838	.774	1
	Sig. (2-tailed)	.018	.018	.018	.368	.437	
	N	3	3	3	3	3	3

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

Figure 6.21: Prototype correlation test

7

Discussion and Conclusion

This chapter discusses the main finding of this research and answering the research questions. Moreover, this chapter also discusses the impact of this research that could contribute to academic and industrial practices. This chapter also discusses the validity of the test and provides suggestions for future work.

7.1. Result summary

This section answered all the research questions that were defined in Section.1.2.1, and in the end answering the main research question.

RQ1. What log structures that are able to facilitate the EA conversion?

The log structure is based on the event log, or it can be said that the structure is an extension of the event log. We based the structure on XES schema, and it is straightforward to use the schema since it is standardised and extensible. Thus the structure was derived from the schema with additional fields that serve the purpose for the EA mining. The event log has two types of data, historical data, and running data. Both can serve the same purpose of this research, the difference between these types is how up-to-date that the data can be. Since the running data is extracted from a running system and historical is from historical data. Moreover, historical data can be used to create a predictive model that can be applied to the running data. Nevertheless, both historical and running data are capable of describing the processes that occur in the enterprise, and it can also store related information to those processes, which in the end can serve the purpose of the structure and create an EA model.

Comparing to the other research in automated EA model documentation, the data source that the others used is limited to certain products or systems [9, 22, 42]. The data sources for our research can be varied and not limited to certain produce or systems since we used the event log as an intermediary between multiple data sources. This is a point that differs our research to others, and this research can be an alternative to automate an EA model. Since we did not do the comparison between our research to others transformation approaches, we can not provide the effectiveness and usability of this research comparing to other approaches. However, we can argue that the process mining not only limited to creating a process model but also with a simple adjustment it can also create architectures that the enterprise has. Thus, this research extending the current process mining research and provide prospect to combining the process observation with the enterprise architecture.

Another important point of this research is how extensive that the data gathering can be since we did not explore in-depth to the event log's Extraction, Transformation, Loading (ETL) processes. Thus, we did not know the operability, usability, and the quality of the

data gathering. Thus, it can render our research useless, since anybody can argue that the data gathering can out weight the transformation process. However, in our defence, there is some research that tried to solve this data gathering process [17, 31].

In addition, using the guideline, we are able to create a log, event though it is simplistic and very limited in the concrete implementation. It purposes only to provide some ground rule to create the log and how to populate the log based on the log structure that we proposed. However, we think this simple guideline can be helpful for anybody that tried to replicate our research and to create an event of their own. Furthermore, since the guideline only limited to one business process that has a workflow, what if the case if an enterprise wants to observe multiple business model that it has. In that case, we suggest that each event log produce an EA model and in the end combining all EA models into one cohesive model.

The event log can contain measurable information regarding the performance of the workflow and its surrounding elements. Using this information we can analyse the performance of the EA model more thoroughly. In this research we analyse using simple mathematical computation, the possibility to incorporate more advance techniques such as EA quantitative analysis [24] is wide open. Thus, not only our research provides a conversion mechanism but also it can be extensible to analyse the performance of an EA model, and it contributes to more robust and automated analysis.

RQ2. What are conversion methods to process logs into EA models?

In the process mining, there are multiple control-flow perspectives that usable for discovering the processes within the systems. In our research, we used heuristic miner as our representative of control-flow perspective. The heuristic miner is simple to use and has the capability to detect noise, incompleteness, and rare workflow. Although in this research we did not fully utilise the capability of noise detection and other functionalities. But the EA mining can produce the EA model that we expected. We also tested the perceived business model that an enterprise has with the reality based on the log, and the result is satisfactory. However, we perceived that if our approach is to be implemented in real life condition, there is a possibility that might be a disparity between what the enterprise perceived to what actually happens since the capability human do a repetitive task cannot be compared to the strength of a machine do things. Hence, our approach will give accurate information regarding the latest condition of the EA of the enterprise.

Another point that we need to highlight is the process model that we used as the comparison based on EA elements. We used the dependency graph to denotes the result of the process model. The graph provides a simple definition and annotation. The more complex graph such as petri-net, process tree can give detailed information regarding the flow of processes, such as join condition of a workflow. However, since the target of our research is to detect components of EA, we think it is not necessary to have a detailed version of a workflow. We also think in the future research we can simplify the spaghetti-like business process (Fig.6.13) into a more simple business process. But, we cannot eliminate the business process, since it is the core of the EA mining and the line of thinking of EA mining is within the workflows that an enterprise have.

7.2. Contributions

This section describes the contribution of this research to academic or industry practices.

- **Process mining can also be extended to generate EA models**

  Currently process mining creates multiple process model each describes the observational result per perspective, from the most common perspective to the least one. Each process model can be varied depending on the objective that users wanted. This research provides an additional model that can be used for the purpose of observation,

instead of process model this research provides EA models. Using the basic principle of the control-flow, this research expands the perspective by adding additional perspective and collect it into one model that can describe the condition of enterprises. It also borrows the capabilities of the process mining: to discover reality using the data, to conform the user perception with reality. In the end, this research provides a new model that can be generated by so-called "EA mining" and it also adapts the features of process mining: reality discovery with conformance to user perception.

- **Contribution to problem statements**

  In the beginning there are three big the problem statements, which are: (1) Manual maintenance with low automation, (2) Maintenance can not cope up with enterprises' growth (3) Ivory tower syndrome. This research can help to contribute to these problems by automating EA models generation, this will help to reduce the manual process (1) and improve the maintenance process (2). This research will also help to reduce disconnectivity between user perception and reality. This research based on the EA model generation on data, whether the data is realtime or historical. Thus, the model that is created will closely resemble the reality in practices. This will reduce the effect of the ivory tower syndrome.

- **Log structure is incorporated from multiple data source**

  The current automation process is limited to certain tools, and data sources [9, 22, 42]. By implementing a centralised log, it is possible to incorporate multiple data sources into a single log file. This also aligns with the nature of the event log, that needed multiple data source to generate the log. In practice, it also makes sense since a system can have interaction with other systems, to collaborate and give certain functions to help the enterprise. These systems can have their own datasource. Thus, this research gave the glimpse of this collaboration (Section.6.1) and able to provide a simple example to combine all the information into a centralised log. In the end, this research will help the enterprise to create a centralised log that cen be used for mining information, especially for EA mining.

- **Analyse function for EA model analysis**

  In addition, event logs also extensible for adding relevant information regarding systems. This feature has enabled a user to add information that can describe the performance of the systems. This research provides a function to analysis EA models using that information. This shown the capabilities of this approach to analyse EA models with multiple analysis functions and multiple analysis measurements. This features will help users not only to generate EA models but also to identify the performance of the EA models.

7.3. Validity

This research test the artefact using a controlled experiment, with a use case that was specifically made for the purpose of this research. However, using other cases, for example, hospital outpatient workflow, or manufacturing process as stimuli to the artefact, more or less resulting similar output with the test that we were conducted since workflows and its information is also available to other systems and business models. Thus, more or less the single case mechanism experiment is able to produce similar output regardless of the input that was inserted.

In comparison between conversion pattern and process models that were created using process mining tools, it can be seen that there is the exact same sequence of flow with the number of frequency and dependency factor. Using this information, we can argue that the conversion pattern successfully implemented the heuristic miner algorithms that was proposed. However, the test that was conducted using manual processes. Since there are no automation tools to compare results that were produced by PM tools to EA models. The test

took a sampling of certain workflow that happens between EA models and process models, this approach was taken since the results that both tools produced were too complex.

In the validation, this research implemented EA mining and created a prototype. The prototype was developed using C#. The research also conducted a performance test to monitor the performance of the prototype against multiple real life datasets. Based on the test, there is a positive and negative correlation between the time needed to execute the prototype with the number of fields that the datasets have. However, different algorithms and code implementation can lead to different results. Thus, the result is just information about how the current prototype can fare against a certain amount of datasets, and the results can not be used as a benchmark.

7.4. Limitations and future work

There are limitations of this research. First of all, EA mining is tighly coupled with events. Thus all the information that describes the enterprise must be connected to the events. This can be problematic when we want to develop a more complex architecture. As can be seen in Archimate, all the constructs can be interconnected with each other not necessarily need to be connected to certain elements. While this research all elements must be connected to the business process. In future work, we can expand the idea of this thesis and create a log that is not bounded to an event, thus removing this limitation and create a new possibility to produce EA models that more complex and complete.

This research used one validation mechanism, which is the single-case mechanism experiment. This experiment is very limited, and all variables were controlled. In the future, we need to test this research with more test such as Technical Action Research [45], introducing users to the artefact. or UTAUT model, predicting the acceptance of the artefact to industrial practices. Thus, by adding more test we can collect more data, and the artefact can be further improved. In the end, it can be published for industrial practices and helps the practitioner to maintain EA model automatically.

Another limitation of this research was, we did not measure the time and cost to obtain the log. Is the expense can outweight the benefit. In addition, what is the efficient method to obtain the log from multiple data sources. All of these questions need separate research to determine whether this research can be an alternative solution to automated EA modelling documentation.

This research also has a limitation in processing a log. Each log is a representation of one business model. Thus, if an enterprise has more than one business model it needs more than one log depending on how much that the business models are. This research only able to process a log at a time, thus when there are multiple logs, then there are multiple EA models. Hence, to obtain the complete overview of the enterprise we need to combine all the models. In future work, this limitation needs to be solved, and it needs a new mechanism to accept multiple logs and create a single cohesive EA model.

Bibliography

[1] BPI Challenges 2013. 9th international workshop on business process intelligence 2013. URL http://www.win.tue.nl/bpi/doku.php?id=2013:challenge.

[2] BPI Challenges 2014. 10th international workshop on business process intelligence 2014. URL http://www.win.tue.nl/bpi/doku.php?id=2014:challenge.

[3] BPI Challenges 2015. 11th international workshop on business process intelligence 2015. URL http://www.win.tue.nl/bpi/doku.php?id=2015:challenge.

[4] R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1377 LNCS: 469–483, 1998. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84890470685&partnerID=40&md5=6b1646f986d196c756dad381edee33a4. cited By 497.

[5] S. Aier, S. Buckl, U. Franke, B. Gleichauf, P. Johnson, P. Närman, C.M. Schweda, and J. Ullberg. A survival analysis of application life spans based on enterprise architecture models. 2009. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84874276834&partnerID=40&md5=b051c4f8e7ac964746169377050f4103. cited By 25.

[6] B. Akman and O. Demirörs. Applicability of process discovery algorithms for software organizations. 2009. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-74549134602&doi=10.1109%2fSEAA.2009.87&partnerID=40&md5=b0155eba45a3a927266f8c872623e91a. cited By 13.

[7] A. Appice, M. Di Pietro, C. Greco, and D. Malerba. Discovering and tracking organizational structures in event logs. 2016. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84971525414&doi=10.1007%2f978-3-319-39315-5_4&partnerID=40&md5=7d22f7b9c7dd8b07190c2294a5113fb9. cited By 3.

[8] M. Bozkaya, J. Gabriels, and J.M. Van Der Werf. Process diagnostics: A method based on process mining. 2009. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-63849202776&doi=10.1109%2feKNOW.2009.29&partnerID=40&md5=9550148820ed7a877153d02d36a00a7d. cited By 41.

[9] M. Buschle, M. Ekstedt, S. Grunow, M. Hauder, F. Matthes, and S. Roth. Automating enterprise architecture documentation using an enterprise service bus. 2012. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84878088713&partnerID=40&md5=b71900fd25a139e3474f21194bed8ab4. cited By 28.

[10] M. Buschle, H. Holm, T. Sommestad, M. Ekstedt, and K. Shahzad. A tool for automatic enterprise architecture modeling. 2012. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84861175015&doi=10.1007%2f978-3-642-29749-6_1&partnerID=40&md5=9c5a0352c2aa50d25accd6096ed107ee. cited By 12.

[11] A. Chuvakin, K. Schmidt, and C. Phillips. *Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management.* 2012. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84902030868&doi=10.1016%2fC2010-0-65241-2&partnerID=40&md5=4b828640a4dab1ece7589f36912ee3cf. cited By 7.

[12] M. Farwick, B. Agreiter, R. Breu, S. Ryll, K. Voges, and I. Hanschke. Automation processes for enterprise architecture management. 2011. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-80054827511&doi=10.1109%2fEDOCW.2011.19&partnerID=40&md5=69fe6213cdeb15e3e60b32e083026201. cited By 26.

[13] M. Farwick, R. Breu, M. Hauder, S. Roth, and F. Matthes. Enterprise architecture documentation: Empirical analysis of information sources for automation. 2013. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84875495802&doi=10.1109%2fHICSS.2013.200&partnerID=40&md5=41f5a6abf0c26feb89a648cf3d57cbbd. cited By 23.

[14] Matthias Farwick, Berthold Agreiter, Ruth Breu, Steffen Ryll, Karsten Voges, and Inge Hanschke. REQUIREMENTS FOR AUTOMATED ENTERPRISE ARCHITECTURE MODEL MAINTENANCE A Requirements Analysis based on a Literature Review and an Exploratory Survey. In Zhang, R and Cordeiro, J and Li, X and Zhang, Z and Zhang, J, editor, *ICEIS 2011: PROCEEDINGS OF THE 13TH INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS, VOL 4*, pages 325–337. 2011.

[15] feedough. Amazon business models. URL https://www.feedough.com/amazon-business-model/.

[16] Gartner. Gartner it glossary. https://www.gartner.com/it-glossary/enterprise-architecture-ea/, 2018. Accessed: 2018-11-30.

[17] E. González López de Murillas, H.A. Reijers, and W.M.P. van der Aalst. Connecting databases with process mining: a meta model and toolset. *Software and Systems Modeling*, pages 1–39, 2018. doi: 10.1007/s10270-018-0664-7. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85042203202&doi=10.1007%2fs10270-018-0664-7&partnerID=40&md5=b127b46961e916918ae78d38e7527f37. cited By 2; Article in Press.

[18] The Open Group. Archimate® 3.0.1 specification. http://pubs.opengroup.org/architecture/archimate3-doc/, 2018. Accessed: 2018-11-30.

[19] The Open Group. Togaf® standard, version 9.2. http://pubs.opengroup.org/architecture/togaf9-doc/arch/, 2018. Accessed: 2018-11-30.

[20] Christian W. Günther and Eric Verbeek. First xes standard definition version 2.0. http://www.xes-standard.org/_media/xes/xesstandarddefinition-2.0.pdf, 2014. Accessed: 2018-08-30.

[21] M. Hauder, F. Matthes, and S. Roth. Challenges for automated enterprise architecture documentation. 2012. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84868312820&doi=10.1007%2f978-3-642-34163-2_2&partnerID=40&md5=833eb2c735baa4dc8138184fa7ad5674. cited By 18.

[22] H. Holm, M. Buschle, R. Lagerström, and M. Ekstedt. Automatic data collection for enterprise architecture models. 2014. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84899904065&doi=10.1007%2fs10270-012-0252-1&partnerID=40&md5=014a51ab7fb1d7a172d6d5cd9fbb9100. cited By 15.

[23] Stephanie Houde and Charles Hill. What do prototypes prototype? In *Handbook of human-computer interaction*, pages 367–381. Elsevier, 1997.

[24] Maria-Eugenia Iacob and Henk Jonkers. Quantitative analysis of enterprise architectures. In *Interoperability of Enterprise Software and Applications*, pages 239–252. Springer, 2006.

[25] P. Johnson, M. Ekstedt, and R. Lagerstrom. Automatic probabilistic enterprise it architecture modeling: A dynamic bayesian networks approach. 2016. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84992603280&doi=10.1109%2fEDOCW.2016.7584351&partnerID=40&md5=a96bd53e180d52ee7f15c58e57e83a37. cited By 1.

[26] A.A. Kalenkova, W.M.P. van der Aalst, I.A. Lomazova, and V.A. Rubin. Process mining using bpmn: relating event logs and process models. 2017. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84944916586&doi=10.1007%2fs10270-015-0502-0&partnerID=40&md5=17952c31349ea77c0da3c89c99db5eee. cited By 8.

[27] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. Systematic literature reviews in software engineering - a systematic literature review. 2009. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-56649086628&doi=10.1016%2fj.infsof.2008.09.009&partnerID=40&md5=209d0cdfec109a5c5af11592697dcbf2. cited By 690.

[28] Marc Lankhorst. *Enterprise architecture at work: Modelling, communication and analysis*. Springer, 2009.

[29] J. Lismont, A.-S. Janssens, I. Odnoletkova, S. vanden Broucke, F. Caron, and J. Vanthienen. A guide for the application of analytics on healthcare processes: A dynamic view on patient pathways. 2016. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84984671884&doi=10.1016%2fj.compbiomed.2016.08.007&partnerID=40&md5=a7345273b9e9f43602cd6e93942648e2. cited By 6.

[30] R.S. Mans, M.H. Schonenberg, M. Song, W.M.P. Van Der Aalst, and P.J.M. Bakker. Application of process mining in healthcare - a case study in a dutch hospital. 2008. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-77953998588&doi=10.1007%2f978-3-540-92219-3_32&partnerID=40&md5=b4ca7c880b58d3b865e9f162b412aa29. cited By 132.

[31] N.A. Otmani, M. Si-Mohammed, C. Comparot, and P.-J. Charrel. Ontology-based approach to enhance medical web information extraction. *International Journal of Web Information Systems*, 2018. doi: 10.1108/IJWIS-03-2018-0017. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85058465522&doi=10.1108%2fIJWIS-03-2018-0017&partnerID=40&md5=84cbb2d06c884b0794dfae5255be54a4. cited By 0; Article in Press.

[32] E. Rojas, J. Munoz-Gama, M. Sepúlveda, and D. Capurro. Process mining in healthcare: A literature review. 2016. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84964506114&doi=10.1016%2fj.jbi.2016.04.007&partnerID=40&md5=41e498de8034d268fa52af81665d9023. cited By 71.

[33] M. Song and W.M.P. van der Aalst. Towards comprehensive support for organizational mining. 2008. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-56049101373&doi=10.1016%2fj.dss.2008.07.002&partnerID=40&md5=05e5e059e3031787ba6ff37272b93629. cited By 147.

[34] Paul Timmers. Business models for electronic markets. *Electronic markets*, 8(2):3–8, 1998.

[35] Margus Valja, Robert Lagerstrm, Mathias Ekstedt, and Matus Korman. A Requirements Based Approach for Automating Enterprise IT Architecture Modeling Using Multiple Data Sources. In *PROCEEDINGS OF THE 2015 IEEE 19TH INTERNATIONAL ENTERPRISE DISTRIBUTED OBJECT COMPUTING CONFERENCE WORKSHOPS AND DEMONSTRATIONS (EDOCW 2015)*, IEEE International Enterprise Distributed Object Computing Conference Workshops-EDOCW, pages 79–87. 2015. doi: {10.1109/EDOCW.2015.33}.

[36] W. Van der Aalst. *Process mining: Data science in action*. 2016. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84979530122&doi=10.1007%2f978-3-662-49851-4&partnerID=40&md5=2c9677c2cfe2b253453b40c50e86ebb3. cited By 233.

[37] W. Van Der Aalst, A. Adriansyah, A.K.A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J.C. Bose, P. Van Den Brand, R. Brandtjen, J. Buijs, A. Burattin, J. Carmona, M. Castellanos, J. Claes, J. Cook, N. Costantini, F. Curbera, E. Damiani, M. De Leoni, P. Delias, B.F. Van Dongen, M. Dumas, S. Dustdar, D. Fahland, D.R. Ferreira, W. Gaaloul, F. Van Geffen, S. Goel, C. Günther, A. Guzzo, P. Harmon, A. Ter Hofstede, J. Hoogland, J.E. Ingvaldsen, K. Kato, R. Kuhn, A. Kumar, M. La Rosa, F. Maggi, D. Malerba, R.S. Mans, A. Manuel, M. McCreesh, P. Mello, J. Mendling, M. Montali, H.R. Motahari-Nezhad, M. Zur Muehlen, J. Munoz-Gama, L. Pontieri, J. Ribeiro, A. Rozinat, H. Seguel Pérez, R. Seguel Pérez, M. Sepúlveda, J. Sinur, P. Soffer, M. Song, A. Sperduti, G. Stilo, C. Stoel, K. Swenson, M. Talamo, W. Tan, C. Turner, J. Vanthienen, G. Varvaressos, E. Verbeek, M. Verdonk, R. Vigo, J. Wang, B. Weber, M. Weidlich, T. Weijters, L. Wen, M. Westergaard, and M. Wynn. Process mining manifesto. 2012. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84863011087&doi=10.1007%2f978-3-642-28108-2_19&partnerID=40&md5=abcaabb60801e3a8657a9cdb43233791. cited By 378.

[38] W.M.P. Van Der Aalst. A general divide and conquer approach for process mining. 2013. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84892496768&partnerID=40&md5=431f8b590792a49b1aa65761df72325a. cited By 23.

[39] W.M.P. Van Der Aalst and M. Song. Mining social networks: Uncovering interaction patterns in business processes. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3080: 244–260, 2004. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-35048882847&partnerID=40&md5=6d8f6a7076153e5104962c1614a3c423. cited By 130.

[40] W.M.P. van der Aalst, H.A. Reijers, A.J.M.M. Weijters, B.F. van Dongen, A.K. Alves de Medeiros, M. Song, and H.M.W. Verbeek. Business process mining: An industrial application. 2007. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-34147190948&doi=10.1016%2fj.is.2006.05.003&partnerID=40&md5=3684b62f49e6b681f9ccc797a9636359. cited By 437.

[41] Bas Van Der Raadt, Raymond Slot, and Hans Van Vliet. Experience report: assessing a global financial services company on its enterprise architecture effectiveness using naomi. In *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, pages 218b–218b. IEEE, 2007.

[42] R. Van Langerak, J.M.E.M. Van Der Werf, and S. Brinkkemper. Uncovering the runtime enterprise architecture of a large distributed organisation a process mining-oriented approach. 2017. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85021230951&doi=10.1007%2f978-3-319-59536-8_16&partnerID=40&md5=1a0abfcbfa42aa57ba0a5dde042859eb. cited By 0.

[43] P. Weber, B. Bordbar, P. Tino, and B. Majeed. A framework for comparing process mining algorithms. 2011. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-79957987064&doi=10.1109%2fIEEEGCC.2011.5752616&partnerID=40&md5=225e7168eeceb63b1aba8ff0ba34fb6a. cited By 9.

[44] AJMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166: 1–34, 2006.

[45] R.J. Wieringa. *Design science methodology: For information systems and software engineering*. 2014. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84956782190&doi=10.1007%2f978-3-662-43839-8&partnerID=40&md5=d0be3688233f4c35d77b8cd2741327ea. cited By 131.

[46] Katharina Winter, Sabine Buckl, Florian Matthes, and Christian M Schweda. Investigating the State-of-the-Art in Enterprise Architecture Management Methods in literature and Practice. *MCIS*, 90, 2010.

[47] Yu Xiao and Maria Watson. Guidance on conducting a systematic literature review. *Journal of Planning Education and Research*, 39(1):93–112, 2019.

A

Automated EA documentation fields mapping

Context	Source (Fields)	Target (Archimate's Elements)	Layer
PAIS [42]	Activity	Business Process	Business
PAIS [42]	Resource	Business Actor, Application Component	Business, Application
Network monitoring [22]	User Account	Business Actor	Business
Network monitoring [22]	Application protocol	Infrastructure interface	Technology
Network monitoring [22]	Application server	System software (end-point)	Technology
Network monitoring [22]	Application Client	Application component	Technology
Network monitoring [22]	Operating System	System software (OS)	Technology
Network monitoring [22]	IP, MAC Address	Device	Technology
Network monitoring [22]	A range of IP address	Network	Technology
ESB [9]	Party	Business Actor	Business
ESB [9]	data type	Business Object	Business
ESB [9]	adapter	Representation	Business
ESB [9]	data type	Data Object	Application
ESB [9]	enterprise service interface	Application Interface	Application
ESB [9]	software component, software product	Application Component	Application
ESB [9]	sofware configuration	Application Collaboration	Application
ESB [9]	computer system	Node	Technology
ESB [9]	system landscape directory	System Software	Technology
ESB [9]	outbound and inbound interface	Communication Path	Technology

75

B

Process discovery algorithms based on literature review

Type	Algorithm	Description
Control Flow	Alpha Miner [32, 36]	One of the first algorithm to detect concurrency, enable to generate process models.
Control Flow	Fuzzy Miner [8, 29, 30, 32]	Enable to genare process models from unstructured behaviours and huge number of activities.
Control Flow	Genetic Miner [26, 32, 36]	Generate process models using evolutionary approah.
Control Flow	Heuristics Miner [30, 32, 36, 40]	Create process models with consideration of frequencies of events and sequences and able to deal with noise and incompleteness.
Control Flow	Inductive Miner [26, 32, 36]	Highly extendible and able to handle infrequent behaviour and deal with huge models and log.
Control Flow	Region-Based Miner [36]	Synthesis of state-based regions and language-based region, resulting ability to generate process models from a transition system or a prefix-closed language.
Control Flow	Trace Clustering [30, 32]	Enable partioning of the event logs to generate simpler and more structured process models.
Organisational	Organizational Miner [8, 36]	Deriving the organisational model from event logs.
Organisational	Resource behavior [36]	Correlating activities, organisational entities, and resources to gain insight about resource behaviour.
Organisational	Resource community tracking [7]	Discover organisational structure using stream data.
Organisational	Role hierarchy miner [29]	Extract information of resources that performing certain activities.
Organisational	Social Network Miner [8, 29, 30, 32, 36, 40]	Extract interpersonal relationship between originators.
Data / Case, Performance / Time	Dotted chart [8, 29, 30, 36]	Representation of Events using dotted chart and express the data's attributes in two dimensional plane.

C

Archi Conversion

```
public static void Export(String exportPath, RelationshipTable rTable, ElementTable eTable, bool isAlg3Enabled)
        {
            //String filename = "C:/Users/root/Desktop/test.archimate";

            String filename = exportPath + String.Format("/EA_MODEL_{0}.archimate", DateTime.Now.ToString("yyyyMMdd_HHmm")) ;

            folder[] folds = new folder[] { };
            folder fold = new folder();
            folderElement[] elements = new folderElement[] { };
            folderElement element = new folderElement();
            folderElementChild[] children = new folderElementChild[] { };
            folderElementChild child = new folderElementChild();
            folderElementChildBounds bound = new folderElementChildBounds();
            folderElementChildSourceConnection[] connections = new folderElementChildSourceConnection[] { };
            folderElementChildSourceConnection connection = new folderElementChildSourceConnection();

            // +++ BUSINESS LAYERS +++
            // registering business process
            elements = new folderElement[] { };
            var businessElements = eTable.ElementRows.Where(w => w.Type.Contains("Business")).Select(s => s);
            Array.Resize(ref elements, businessElements.Count());

            int i = 0;
            foreach (var item in businessElements)
            {
                element = new folderElement();
                element.id = item.GUID_Element;
                element.name = item.ElementAlias;
                element.type = TypeConversion(item.Type);
                elements[i] = element;
                i++;
            }

            // add business actor to folder
            fold = new folder();
            fold.id = Guid.NewGuid().ToString();
            fold.name = "Business";
            fold.type = "business";
            fold.element = elements;
            Array.Resize(ref folds, folds.Length + 1);
            folds[folds.Length - 1] = fold;

            // +++ APPLICATIONS LAYERS +++
            elements = new folderElement[] { };
            var appElements = eTable.ElementRows.Where(w => w.Type.Contains("Application")).Select(s => s);
            Array.Resize(ref elements, appElements.Count());

            i = 0;
            foreach (var item in appElements)
            {
                element = new folderElement();
                element.id = item.GUID_Element;
                element.name = item.Element;
                element.type = TypeConversion(item.Type);
                elements[i] = element;
                i++;
            }

            // add business actor to folder
```

```
fold = new folder();
fold.id = Guid.NewGuid().ToString();
fold.name = "Application";
fold.type = "application";
fold.element = elements;
Array.Resize(ref folds, folds.Length + 1);
folds[folds.Length - 1] = fold;

// +++ RELATIONSHIP +++
Dictionary<String, String> relationships = new Dictionary<string, string>();

// registering business process
elements = new folderElement[] { };
Array.Resize(ref elements, rTable.RelationshipRows.Count);

i = 0;
foreach (var item in rTable.RelationshipRows)
{
    element = new folderElement();
    element.id = item.GUID_Relationship;
    if (item.Frequency > 0 || item.Dependency > 0.0)
    {
        //element.name = item.Frequency + ", " + item.Dependency;
        //element.name = item.Frequency + "";
    }
    element.type = TypeConversion(item.Relationship);
    element.source = FindGuid(item.Source, eTable);
    element.target = FindGuid(item.Target, eTable);
    elements[i] = element;
    i++;
}

// add relationship to folder
fold = new folder();
fold.id = Guid.NewGuid().ToString();
fold.name = "Relations";
fold.type = "relations";
fold.element = elements;
Array.Resize(ref folds, folds.Length + 1);
folds[folds.Length - 1] = fold;

// +++ DIAGRAM +++
children = new folderElementChild[] { };
Array.Resize(ref children, eTable.ElementRows.Count + 1);

i = 0;
foreach (var iEl in eTable.ElementRows)
{
    connections = new folderElementChildSourceConnection[] { };
    child = new folderElementChild();
    child.id = iEl.GUID_Diagram;
    child.archimateElement = iEl.GUID_Element;
    child.type = "archimate:DiagramObject";
    bound = new folderElementChildBounds();
    bound.height = (ushort)ElementBound.HEIGHT;
    bound.width = (ushort)ElementBound.WIDTH;
    if (iEl.Type == Constants.ArchimateElementType.BUSINESS_PROCESS && isAlg3Enabled)
    {
        bound.width = 200;
    }
    bound.x = (ushort)iEl.xPos;
    bound.y = (ushort)iEl.yPos;

    child.bounds = bound;

    // set-up connections
    if (iEl.Position == ElementPosition.END)
    {
        var listRelationship = rTable.RelationshipRows.Where(s => s.Target == iEl.Element).Select(s => s);
        string conn = "";
        foreach (var iRel in listRelationship)
        {
            conn += iRel.GUID_Connection + " ";
        }
        child.targetConnections = conn;
    }
    else
    {
        var listRelationship = rTable.RelationshipRows.Where(s => s.Source == iEl.Element).Select(s => s);
        Array.Resize(ref connections, listRelationship.Count() + 1);

        int j = 0;
        foreach (var iRel in listRelationship)
        {
```

```
                connection = new folderElementChildSourceConnection();
                connection.type = "archimate:Connection";
                connection.id = iRel.GUID_Connection;
                connection.source = iEl.GUID_Diagram;
                var target = eTable.ElementRows.Where(s => s.Element == iRel.Target).Select(s => s).First();
                connection.target = target.GUID_Diagram;
                connection.archimateRelationship = iRel.GUID_Relationship;
                connections[j] = connection;

                if (iEl.Position != ElementPosition.START)
                {
                    var lastSource = rTable.RelationshipRows.Where(s => s.Target == iRel.Source).Select(s => s);
                    string conn = "";
                    foreach (var item in lastSource)
                    {
                        conn += item.GUID_Connection + " ";
                    }
                    child.targetConnections = conn;
                }
                j++;
            }

            child.sourceConnection = connections;
        }

        children[i] = child;
        i++;
    }

    element = new folderElement();
    element.id = Guid.NewGuid().ToString();
    element.type = "archimate:ArchimateDiagramModel";
    element.name = "EA miner results";
    element.child = children;
    elements = new folderElement[1] { element };

    // add view to folder
    fold = new folder();
    fold.id = Guid.NewGuid().ToString();
    fold.name = "Views";
    fold.type = "diagram";
    fold.element = elements;
    Array.Resize(ref folds, folds.Length + 1);
    folds[folds.Length - 1] = fold;

    // +++ FINALISE XML +++
    model xml = new model();
    xml.id = Guid.NewGuid().ToString();
    xml.version = "4.0.0";
    xml.name = "test";
    xml.folder = folds;

    using (TextWriter writer = new StreamWriter(filename))
    {
        XmlSerializer serializer = new XmlSerializer(typeof(model));
        serializer.Serialize(writer, xml);
    }
}

private static String FindGuid(String element, ElementTable eTable)
{
    if (eTable != null )
    {
        return eTable.ElementRows.Where(w => w.Element == element).Select(s => s.GUID_Element).First();
    }
    return "";
}

private static String TypeConversion(String type)
{
    if (type == Constants.ArchimateElementType.BUSINESS_ACTOR)
    {
        return "archimate:BusinessActor";
    }
    else if (type == Constants.ArchimateElementType.BUSINESS_PROCESS)
    {
        return "archimate:BusinessProcess";
    }
    else if (type == Constants.ArchimateElementType.APP_COMPONENT)
    {
        return "archimate:ApplicationComponent";
    }
    else if (type == Constants.ArchimateRelationship.SERVING)
    {
```

```
            return "archimate:ServingRelationship";
        }
        else if (type == Constants.ArchimateRelationship.ASSIGNMENT)
        {
            return "archimate:AssignmentRelationship";
        }
        else if (type == Constants.ArchimateRelationship.TRIGGERING)
        {
            return "archimate:TriggeringRelationship";
        }
        return "";
    }

}
```