UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Shefali Ajit Sharma

Waste Identification from Event Logs

Master's Thesis (30 ECTS)

Supervisor: Fredrik Milani

Tartu 2021

Waste Identification from Event Logs

Abstract:

Organizations execute a variety of business processes to meet their business objectives. Therefore, they seek to constantly improve such processes. One way to improve the efficiency of processes is to identify and eliminate wastes in a process. Analysts use different process mining software to discover and analyze business processes. Event logs, i.e., data captured from the execution of business processes, are used to discover and analyze processes to identify wastes. To identify wastes from event logs, analysts need to know exactly what to look for. However, wastes are manifested in business processes in different ways. Therefore, manifestations of wastes that the analyst is unfamiliar with, remain hidden. This thesis aims at identifying the manifestations of wastes in business processes and how to detect them from event logs. To this end, 187 relevant papers were identified and subjected to content analysis. From these, manifestations of 8 wastes in business processes were elicited. Following this, a framework for how to detect such wastes from business process event logs was derived. Thus, the contribution of the thesis is a framework for identifying wastes from event logs.

Keywords:
Waste, Process mining, Event log, Waste identification, Waste detection, Business process

CERCS:
P170 Computer science, numerical analysis, systems, control

Raiskamise tuvastamine sündmuslogidest

Lühikokkuvõte:
Organisatsioonid täidavad oma äriliste eesmärkide saavutamiseks mitmeid erinevaid äriprotsesse. Sellest tulenevalt otsivad organisatsioonid pidevalt võimalusi selliste protsesside parendamiseks. Üks võimalus protsesside efektiivsuse tõstmiseks on võimaliku protsessis esineva raiskamise tuvastamine ja kõrvaldamine. Analüütikud kasutavad äriprotsesside tuvastamiseks ja analüüsimiseks erinevaid protsessikaeve rakendusi. Protsesside analüüsi ja raiskamise tuvastamise sisendiks on sündmuslogid, mis sisaldavad protsessi täitmise käigus kogutud andmeid. Raiskamise sündmuslogidest tuvastamiseks peavad analüütikud teadma täpselt millele tähelepanu pöörata. Samas raiskamine ilmneb äriprotsessides mitmel erineval moel ning need raiskamised, mille ilmingutega analüütik ei ole tuttav, jäävad analüütiku eest peidetuks. Käesoleva lõputöö eesmärk on tuvastada äriprotsessides esinevate raiskamiste ilmingud ning pakkuda lahendused nende ilmingute sündmuslogidest avastamiseks. Sellel eesmärgil tuvastati 187 antud teemaga seonduvat artiklit ning teostati nende sisuanalüüs. Nende artiklite põhjal toodi esile 8 raiskamise tüüpi ning seejärel koostati nende raiskamise tüüpide tuvastamiseks sobiv raamistik. Seega on käesoleva lõputöö panuseks raiskamise sündmuslogidest tuvastamise raamistik.

Võtmesõnad:
Raiskamine, Protsessikaeve, Sündmuslogi, Raiskamise tuvastamine, Raiskamise avastamine, Äriprotsessid

CERCS:
P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

1 Introduction

Organizations manage several business processes to achieve its goals [1]. A business process is a collection of activities, events and decision points that involve multiple actors and objects, which collectively lead to an outcome that is of value to an organization [1]. However, as Hammer put it: "every good process eventually becomes a bad process" [2]. For example, Uber's rapid growth had resulted in high process variation leading to a lack of consistency on how the customer support issues were handled [3]. This lack of consistency created additional process waste [3]. Thus to maintain the quality of the process, every good process needs to be continuously managed so to be improved with various business process management techniques.

Dumas et al. [1] defined Business Process Management (BPM) as "a body of methods, techniques and tools to identify, discover, analyze, redesign (improvement), execute and monitor business processes in order to optimize their performance" . Process redesign helps to identify the changes needed to improve the process. However, it is crucial to analyze the current business process to discover various improvement opportunities. One of the aspects analyzed to improve business processes is that of waste. Wastes are instances in a business process that does not add any value from the customer's perspective [4, 5]. Therefore, the main aim of waste analysis is to identify such non-value added parts of a business process [1].

Toyota introduced the concept of lean manufacturing that aims at continuous improvement by delivering value and eliminating waste [6]. The waste analysis process is one of the key techniques of the Toyota Production System and has been integrated into various management paradigms such as lean management [7]. Ohno classified wastes into seven categories: overproduction, waiting, transportation, overprocessing, inventory, unnecessary motion, and defects [6]. Each of these wastes is expressed in different forms when considering different processes [4, 5]. However, the identification of these wastes requires manual effort. One of the widely used tools to identify and reduce waste is Value Stream Mapping (VSM) [8]. VSM is a pencil and paper tool that helps to understand the current value-adding production flow. Rother et al. [8] followed the production path from customer to supplier and created a visual representation of the material and information flow process . To overcome the limitations of VSM, Carvalho et al. [9] proposed a Waste Identification Diagram (WID) to represent a complex production unit and identify different types of wastes. In both these techniques, the data is collected manually through observations and interviews. Therefore, they are more prone to human error. Considering the capability of modern IT systems to log data, there is a potential to use such data for identifying wastes in business processes.

Modern IT systems that support process executions record events corresponding to the execution of a task. All these event records can be extracted from the system's database

and represented as an event log. An event log is a collection of timestamped event records and each event record indicates the execution of a task in a process [1]. Process mining analyzes the performance and conformance of business processes based on the event logs produced during their execution [1]. Process analysts use different process mining tools such as Apromore, Celonis, ProM and Disco to analyze the process in order to identify sources of inefficiencies. For example, a process analyst can use Apromore to see the waste in the form of bottlenecks while replaying the traces of an event log in the log animator or a handover of a task between resources can be seen by applying primary and secondary attribute filters [10]. However, there are no specific rules to help the analysts identify waste from event logs and how they manifest in a process. The analysts rely on analysis templates to identify inefficiencies in a business process. These templates can only be helpful when the analyst knows what to look for. Furthermore, there might be different expressions of waste hidden in the event log that remains undetected.

To automatically discover wastes in a business process from event logs, one needs to identify the different ways the seven wastes manifest themselves in a business process. The primary research objective of the thesis is to identify different data-driven rules to discover wastes in a business process from event logs. Based on the objective and focus areas, the following research questions (RQ) are formulated:

RQ 1: How do wastes manifest themselves in a business process?
RQ 2: How are the manifestations of wastes identified in a business process?
RQ 3: How can manifestations of wastes in business processes be discovered from event logs?

The research questions aim to identify the different manifestations of waste in a business process, the various methods and metrics used to identify waste, and the different ways in which waste can be detected from event logs. The contribution of this thesis is a framework that outlines data-driven rules to identify wastes from event logs. This framework will help the process analysts understand how wastes can be detected in event logs. A combination of deductive and inductive content analysis is performed to answer the research questions. The gathered data is further analyzed, tagged and, structured into different categories. Based on the analysis, different data-driven rules are formulated and presented in the framework.

The rest of the thesis is structured as follows. Section 2 and Section 3 presents the concepts that provide a relevant background and the overview of the related work. Section 4 describes the procedure of the research methodology. Section 5 presents the results of the content analysis approach. The data-driven method and patterns to identify wastes in business processes from event logs are presented in section 6. The results and limitations are discussed in section 7. Finally, section 8 draws the conclusion.

2 Background

This section describes the key concepts that provide a relevant background for the thesis. It covers the concepts of business processes, business process management, lean management waste analysis, and process mining.

2.1 Business Processes

A business usually goes through a series of steps to achieve its goals. Dumas et al. [1] defined a business process as "a collection of inter-related events, activities, and decision points that involve a number of actors and objects, which collectively lead to an outcome that is of value to at least one customer." According to Hammer and Champy [2], a business process is "a collection of activities that takes one or more inputs and creates an output that is of value to the customer". Based on the definitions, a business process encompasses many events, activities, and decision points that may lead to a positive or negative outcome. A business process usually involves actors and objects that can be internal or external to an organization. The internal actors operate inside the organization where the process is executed while the external actors operate outside the organization [1]. All these components, when combined, form the ingredients of a business process. Figure 1 presents the ingredients of a business process.

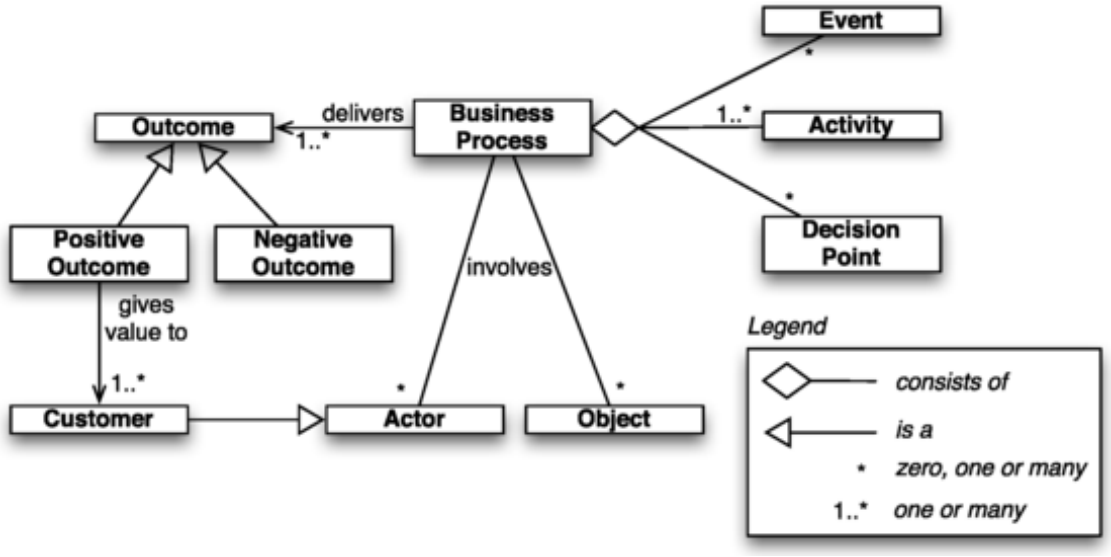

Figure 1: Ingredients of a Business Process [1]

Organizations use business processes to deliver a product or a service to customers. The way a process is designed and performed impacts both the quality and efficiency of the provided service [1]. An organization can outperform another organization if it has a better process execution strategy [1]. It is essential for every organization to constantly monitor and improve a business process to maintain the quality.

2.2 Business Process Management

Dumas et al. [1] defined BPM as "a body of methods, techniques and tools to identify, discover, analyze, redesign (improvement), execute and monitor business processes in order to optimize their performance". BPM can be decomposed into 6 major phases, implemented as a continuous cycle (see Figure 2) to manage the business process [1]. The first step in the BPM lifecycle is process identification. The importance of process

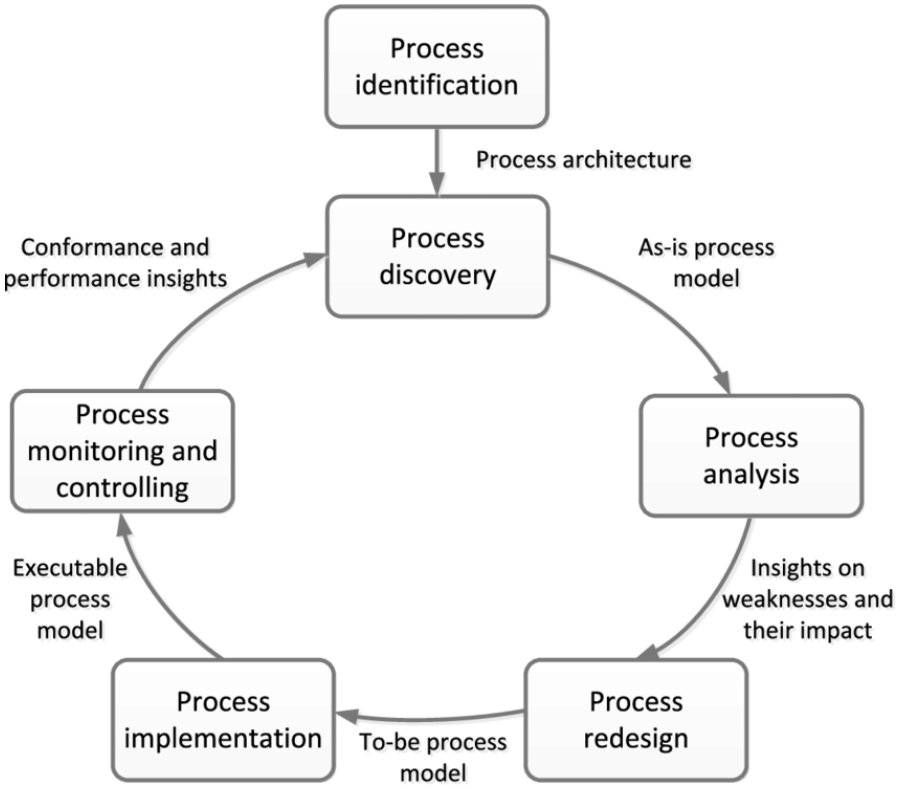

Figure 2: BPM Lifecycle [1]

identification can be understood by looking into the strategic context of an organization. In most organizations, some processes are prioritized more than others [1]. In this phase, processes relevant to an organization are identified. The outcome of this phase is

a new process architecture that provides an overall view of the business processes in an organization [1]. The next phase in the BPM lifecycle is process discovery. In this step, the current state of a process is documented in the form of one or more as-is process models [1]. Dumas et al. [1] introduced the four main process discovery tasks: defining the setting, gathering information, conducting the modeling task, and assuring process model quality. The as-is process models obtained from the previous step are then further analyzed to identify issues associated with it [1]. The process analysis phase aims to identify unnecessary steps in a process, different sources of issues or wastes, and quantify them using different performance metrics [1]. A thorough analysis of a process may pave the way for different redesign opportunities. Different change mechanisms are identified in the process redesign phase to address the issues discovered in as-is process models. The output of this phase is a "to-be" process model [1]. The to-be process model is implemented in the process implementation phase. Process implementation involves two main aspects: organizational change management and automation [1]. Organizational change management consists of activities needed to change the way of working of all participants involved in a process [1]. Automation refers to the development and deployment of IT systems that will support the "to-be" process [1]. Finally, once the redesign process is implemented, relevant data is extracted to obtain insights about the performance of the process [1]. Different corrective actions are then taken to handle various issues identified during the monitoring phase. This requires repeating the BPM cycle [1].

Analyzing a business process plays a crucial role in identifying the different issues in a business process. Qualitative process analysis involves two techniques to identify unnecessary steps in a process and sources of waste [1]. Value Added analysis is one technique that identifies unnecessary steps in a process [1]. The steps within a process that directly contributes to positive outcomes are Value Adding (VA) [1]. For example, consider a process for repairing a washing machine. The steps where the technician detects the problem are value-adding, as they contribute to the outcome the customer desires, which is the machine is repaired [1]. However, some steps do not directly add value to the customer, but they are crucial for the business. These steps are known as Business Value Adding (BVA) [1]. For example, the technician records issues and their fixes into the system. This information does not add any value from the customer's point of view [1]. However, it adds value to the business as it helps the company build a knowledge base to help new technicians in case they face the same issue [1]. The second technique to identify unnecessary steps in a process is waste analysis [1]. Waste analysis can be seen as the reverse of value-added analysis [1]. The main aim of the waste analysis is to identify waste in the process [1]. Waste analysis has been integrated into many lean management paradigms [7]. The focal point of lean management is the continuous improvement of a business process.

2.3 Lean Management and Waste Analysis

The principle focus of Lean is the identification and elimination of non-value adding activities (i.e., waste) to improve the quality of a process [11]. A successful application of lean is identifying value, understanding flow, and characterization of waste [12]. The waste analysis takes a negative angle and is one of the key techniques of the Toyota Production System (TPS) [1]. Ohno classified wastes into seven types: transportation, inventory, motion, waiting, overproduction, overprocessing, and defects [6].Transportation waste deals with any movement of materials from one location to another [13]. For example, in the manufacturing industry, inventory is moved from the warehouse to the site location. Inventory waste expresses itself in the form of Work-in-process (WIP) [13]. For example, early deliveries may result in more stock than needed. Motion waste is the unnecessary movement of people to perform a task [14]. For example, to fix a machine, a worker must move from one location to another to pick up the tools. Waiting waste refers to any delay in the actions that accomplish process transformation [13]. Any delay between the end of one activity and start of the next activity leads to a delay in the completion of the entire process [14]. For example, the delay between the arrival of a truck to pick up goods and loading of the trailer causes a delay in the schedule [14]. Overprocessing and overproduction waste are very closely related. Overprocessing waste refers to the steps that are unnecessarily performed in a process [13]. On the other hand, overproduction waste occurs when an entire process is executed, but it does not add any value on completion [13]. For example, painting parts of a car that a customer will never see is overprocessing waste, and producing the same parts before they are needed is an example of overproduction waste. The waste that occurs out of making defective products is known as defects [13]. Production of defective parts or correction is also considered as defect [13]. Any production that can result in rework/scrap is regarded as a defect [14]. All these seven wastes are considered as symptoms of inefficiencies in a system. Some of these wastes can be traced down to specific steps in a business process while others are hidden between the steps [1]. Dumas et al. [1] grouped the seven wastes into three higher-level categories . The three main categories are:

1. **Move:** This category contains the wastes related to movement: transportation and motion [1].
2. **Hold:** This waste category contains wastes that arise from holding something. The category includes inventory and waiting waste [1].
3. **Overdo:** This category includes the wastes arising from doing more than is necessary to deliver value to the customer. This category includes defects, overprocessing and, overproduction waste [1].

The first type of waste related to movement is transportation. In a business process, physical transportation often occurs when there is a handoff of work between process participants [1]. For example, whenever a document is sent from one process participant to another, a handoff occurs between the process participants [1]. The second type of waste related to movement is motion. Motion waste refers to a process participant moving from one place to another during the execution of a business process [1]. For example, a process participant has to switch from one application to another while performing a task [1]. In a business process, inventory waste shows up in the form of Work-in-Process (WIP) [1]. WIP is the number of cases that have started and have not yet completed [1]. Another type of waste in the hold category is the waiting waste [1]. This occurs when a task is waiting for a process participant or a process participant is waiting for a task [1]. Overprocessing, overproduction, and defects are related to overdoing. While overprocessing waste occurs when an unnecessary task is executed, overproduction waste occurs when an entire process instance is executed and does not add any value upon completion [1]. The last type of overdo waste is defects. In a business process, defect waste includes all the work performed to repair or correct a defect in a process [1]. Defects waste includes rework which refers to a task that has been previously executed in the same case and is performed again due to a defect [1].

Many related work papers discuss different ways of identifying and eliminating waste by mapping the entire process [15–17]. All these techniques are performed manually and rely on human input obtained from observations, interviews, and surveys. Thus, there is a need for a data-driven approach to identify waste. The event logs obtained from BPM and enterprise systems capture the execution of each task in a business process [1]. Process mining helps to extract insights from these event logs.

2.4 Process Mining

Most of the BPM and enterprise systems record events corresponding to an execution of a task. These event records are extracted from the database and represented as an event log [1]. An event log is a collection of timestamped records [1]. Most mining techniques require each event in an event log to have three attributes as a minimum requirement: case identifier, event class, and end timestamp [1]. For example, an event record in an event log captures that Eva has received an order at a given point in time. Process mining techniques help to dig deeper to capture various insights from the event logs. Dumas et al. [1] defined process mining as "a family of techniques to analyze the performance and conformance of business processes based on event logs produced during their execution". As shown in Figure 3, the process mining techniques are classified into four use cases: automated process discovery, conformance checking, performance mining, and variant analysis.

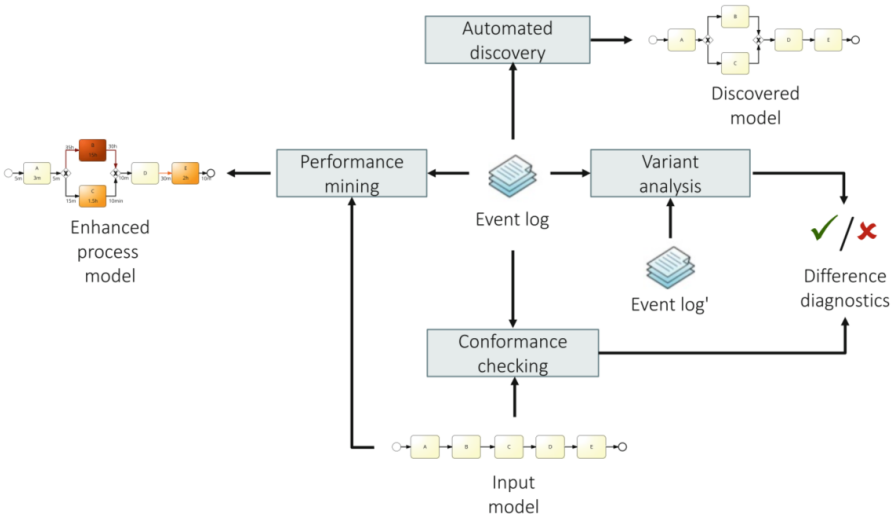

Figure 3: Process Mining Techniques [1]

Automated process discovery produces a process model that matches the observed behavior in an event log [1]. Conformance checking techniques take the generated process model and event log as input and produce a list of differences between the process model and the event log to identify deviations [1]. Performance mining technique produces an enhanced model with additional elements to identify the different pain points in a process [1]. The final use case of process mining is variant analysis. The variant analysis technique takes two event logs as input and produces a list of differences [1]. Process analysts use different process mining tools to analyze the event logs. A few of the widely used tools for process mining are Apromore, Celonis, ProM, and Disco. ProM and Disco are desktop-based applications, while Apromore is a web application. Apromore provides functionalities from automated discovery of process models to real-time predictive monitoring [10]. The use of different process mining techniques allows us to identify various patterns or deviances present in an event log.

3 Related Work

This section presents an overview of the related work concerning the different manifestations of waste in a business process, the different methods or metrics used to identify waste, and how waste can be identified from event logs.

The SLRs on the meaning of waste and business improvement opportunities were identified to understand the concepts of wastes and different identification techniques. Although improvement opportunities are not in the scope of this thesis, different papers included in the SLR on improvement opportunities discussed the different identification techniques before stating the improvement opportunities. In addition to this, more papers were extracted on specific expressions of wastes. Many papers discuss the different ways of identifying wastes [8, 9, 11, 18, 19] but very few present ways of detecting the wastes from event logs [20–24].

The lean philosophy focuses on a continuous process improvement approach and uses various tools and methods to identify and eliminate wastes [25]. One of the commonly used tools to identify wastes is Value Stream Mapping (VSM), and Waste Identification Diagram (WID) [8, 9, 18]. VSM represents the production flows, maps values streams of the product, and helps in identifying different types of waste [8]. Carvalho et al. [18] used VSM and WID to understand the current status of the manufacturing flow-related challenges like overproduction, work-in-process, and inefficient use of man-hours . The VSM+WID helps visualize the gap between the current and future state [18]. Buchanan et al. [19] proposed different solutions to solve the long delay time experienced by the patients in a hospital by mapping the whole patient trail, from referral to discharge of the patient. A combination of interviews and surveys were used to collect the staff opinions, and experience [19]. Alahyari et al. [11] performed an exploratory study of waste in software development organizations by conducting multiple interviews to understand how the organization's people consider waste and how they deal with it.

Delias [22] linked different sources of waste to the different process patterns in an event log of the building permit application process obtained from five Dutch municipalities. Delias [22] mapped the activities performed by a different resource as a handover between the resources, which is a kind of unnecessary transportation. A ping-pong situation where a resource receiving back a case that was handed out to another resource is considered as another example of unnecessary transportation [22]. Delias [22] also linked the execution of activities in batches to inventory, waiting and overproduction waste by considering the cases' duration. Martin et al. [23] identified five types of batch processing from event logs by using a resource-activity centered approach and considering the task or case based orientation. Masn et al. [26] identified bottlenecks in an hospital by searching for instances which indicate a high waiting time in the event log. Verenich et al. [21] minimized overprocessing waste from event logs by ordering

knockout checks at runtime based on predictive machine learning models. Leemans et al. [27] identified frequent patterns by using resource perspective in combination with episode discovery. Rework in an event log expresses themselves in the form of directly-follows or eventually-follows loop [10, 24, 28]. Swennen et al. [24] further classified directly-follows and eventually-follows rework by taking into consideration the resource perspective.

Although the concept of waste has been studied, most of the papers present practices to identify and reduce waste in the form of case studies and methodologies by using manual approaches to identify wastes [4, 8, 9, 29]. Thus, their methods are useful for analysts and can be used as input for data-driven identification of wastes. However, they do not outline how such wastes can be detected from event logs. In this regard, this work is supplementary in that it takes it a step further by considering waste detection from event logs. Some studies extract different patterns from event logs, i.e., consider identification of waste from business processes discovered by process mining techniques [20, 22, 24, 26, 30]. However, such work focuses on one or a set of wastes. This work, on the other hand, considers all seven categories of wastes. Furthermore, in this work, different manifestations of the seven wastes are considered. Thus, although this thesis does not provide software for detecting wastes, it complements research in process mining for waste detection by providing a framework that aid analyst in developing algorithms for automated discovery of wastes from event logs.

4 Methodology

This section presents the method and procedure followed to answer the research questions. The following subsections describe the problem identification and research questions, the data collection and analysis procedure, and the framework elicitation process.

4.1 Problem Identification

The research aims to identify the different types of waste in a business process and how they can be identified from event logs. This involves uncovering the various manifestations of wastes, the different patterns observed in an event log, and mapping the results to the existing predetermined categories, i.e., the seven wastes. Thus, a combination of inductive and deductive content analysis approach [31, 32] was used to conduct in-depth research to synthesize, refine, and map scattered knowledge related to wastes, process mining, and event logs. A deductive content analysis process is performed when a hypothesis is under investigation [31]. This method was used to identify how the seven wastes of lean management manifest themselves in a business process and the different methods, metrics, or rules used to identify the wastes in a business process. The inductive content analysis is an exploratory research method performed when there is no previous knowledge [31]. Additional types of wastes in a business process and the different patterns in an event log were identified using the inductive content analysis approach. There are various ways in which an analyst can pinpoint waste using tools like Apromore, Celonis, ProM, and Disco. For example, we can identify an activity handoff between two resources by using the primary and secondary attribute filter in Apromore [10]. However, there are no specific rules that a process analyst can follow to identify wastes from the event logs.

The blended content analysis approach helped base the findings on the existing literature, thereby confirming and complimenting the previous research by proposing a data-driven method for waste identification from event logs. Figure 4 depicts the procedure followed to answer the research questions. Based on the objective and focus areas, the following research questions (RQ) are formulated:

RQ 1: How do wastes manifest themselves in a business process? This research question aims to identify different ways the seven categories of waste express themselves in a business process.

RQ 2: How are the manifestations of wastes identified in a business process? Numerous papers present practices to identify and reduce waste in the form of case studies and methodologies. This research question aims to identify different ways in which wastes are currently being identified in a business process. This may include different methods, metrics, or rules to identify wastes.

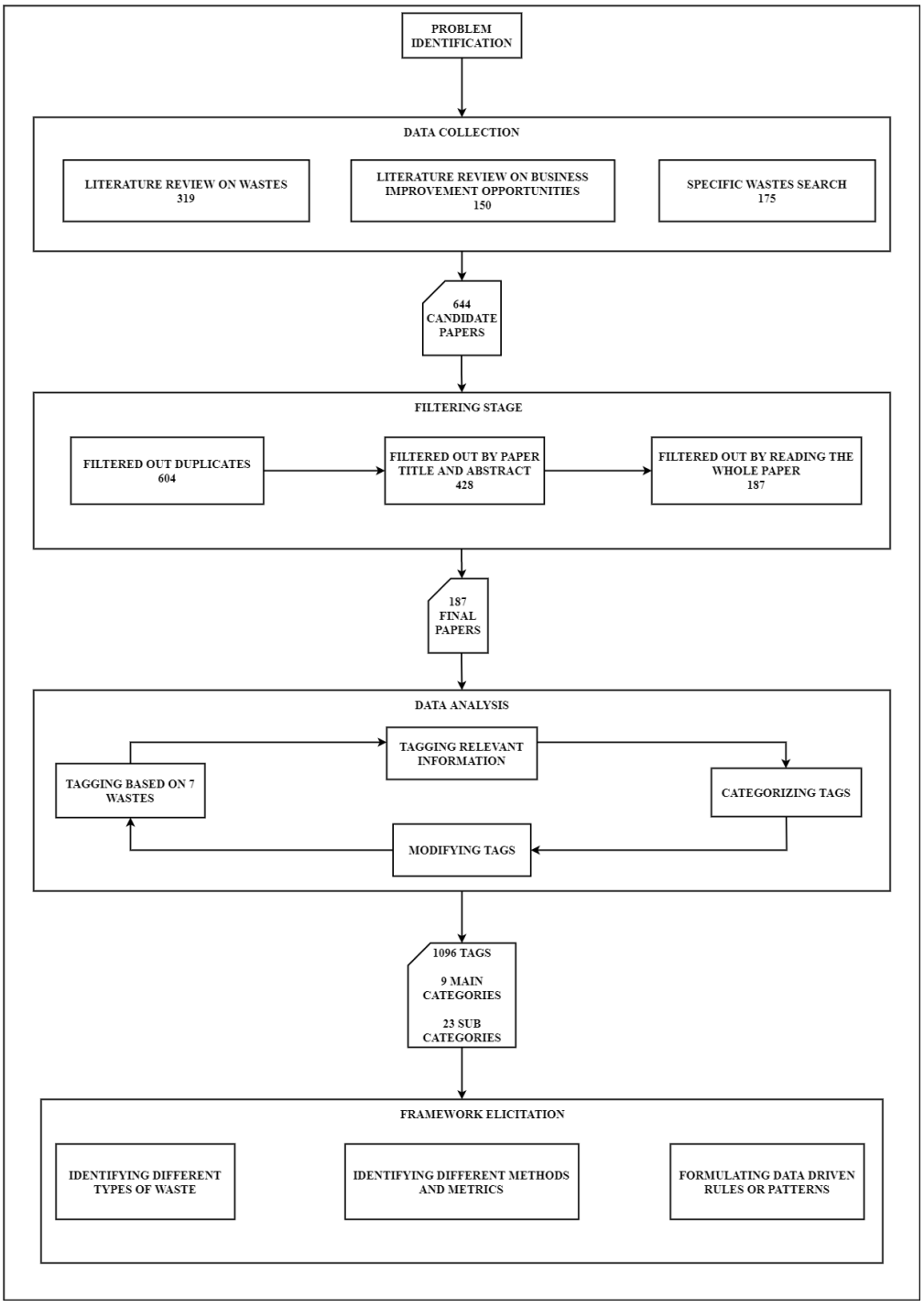

Figure 4: Methodology

RQ 3: How can manifestations of wastes in business processes be discovered from event logs? This research question aims to discover the different ways we can identify wastes from an event log. This will help to uncover different patterns in an event log and associate them with wastes.

4.2 Data Collection

The data collection procedure was executed in compliance with the systematic literature review (SLR) guidelines as proposed by Kitchenham [33]. To understand the relevant waste concepts and different waste identification techniques, we initiated our analysis by searching for the SLR performed on wastes and different business improvement opportunities. Thurer et al. [13] performed an SLR on the meaning of waste to identify the different concepts and use these concepts in the literature related to waste. A total of 319 articles were analyzed by Thurer et al. [13]. Lashkevich [34] performed an SLR to identify different business process improvement opportunities. Even though Laskevich [34] mainly focused on business process improvement opportunities, some papers discussed the different identification techniques before stating the improvement opportunities. A complete list of references was requested from both the authors [13, 34] resulting in 469 papers.

Furthermore, to identify papers that might have been excluded in the filtering process of the previously mentioned SLRs, additional searches of papers were conducted. Therefore, specific forms of wastes in a business process were searched on an individual basis to ensure that relevant articles were not missed. The search was performed on Google Scholar because it contains the search metadata of literature across different publishing formats and disciplines, including grey literature such as non-academic publications. Given the seven wastes, search strings to identify candidate papers were developed based on each type of the seven wastes. The different search strings used to perform the search included the following:

1. ***(("discover" OR "identify") AND ("handoffs" OR "handovers") AND ("process" OR "processes" OR "business process"))*:** This search string identifies papers related to handoffs in a business process.
2. ***(("discover" OR "identify") AND ("rework") AND ("process" OR "processes" OR "business process"))*:** This search string helped us identify research about the different ways a rework is identified in a business process.
3. ***(("discover" OR "identify") AND ("task switching") AND ("process" OR "processes" OR "business process"))*:** This search string identifies research related to different ways in which task switching can be identified in a business process.

4. ***(("discover" OR "identify") AND ("WIP" OR "work in process" OR "work-in-process") AND ("process" OR "processes" OR "business process"))*:** This search string helped us identify the different ways to identify WIP in a business process.

5. ***(("discover" OR "identify") AND ("overprocessing" OR "over-processing") AND ("process" OR "processes" OR "business process"))*:** This search string identifies the different ways over-processing waste occurs in a business process.

6. ***(("discover" OR "identify") AND ("overproduction" AND "waste") AND ("process" OR "processes" OR "business process"))*:** This search string helped us identify the different ways overproduction waste occurs in a business process.

7. ***(("discover" OR "identify") AND ("waiting" OR "delay") AND ("process" OR "processes" OR "business process"))*:** This search string helped us identify the different ways waiting waste occurs in a business process.

The results of each search string were sorted by relevance. The first 25 papers were picked up for further analysis because the rest of the papers presented the different lean methodologies or did not discuss the specific waste in particular. This additional search resulted in 175 papers obtained from all the search strings. In total, we extracted a list of 644 candidate papers.

4.3 Paper Selection Procedure

Following the guidelines by Kitchenham [33], the paper selection procedure aims to identify the research relevant to this thesis. The selection procedure was executed by first filtering the duplicates and then filtering by reading the paper title and abstract. The result of these steps was a total of 428 papers. After reading the entire paper, 187 papers formed the final list of the relevant research study. The selection of relevant study results comprised of the following inclusion and exclusion criteria:

1. **Inclusion Criteria (IC)**

    - **IC1: Is the paper relevant to the domain of identification of wastes in a business process?** The criterion aims to filter out the papers that are not in line with the research objective and scope. For example, studies related to environmental waste, a description of methodologies or principles for identifying waste, and waste elimination are filtered out.

    - **IC2: Does the paper present different identification techniques to identify wastes in a business process?** This criterion helps to filter relevant papers that present different methods and metrics to identify wastes in the form of case studies and methodologies.

- **IC3: Does the paper describe the various patterns observed in an event log?** This criterion helps to identify research related to the different patterns observed in an event log. This will help us to identify different wastes in a business process from event logs.

2. **Exclusion Criteria (EC)**

   - **EC1: Is the full-text version digitally accessible?** Papers accessible via digital libraries subscribed to by the University or available on the Internet in free access are considered accessible. Papers that require payment to be accessed are considered inaccessible.
   - **EC2: Is the paper a duplicate?** Duplicate papers involved those papers that had the same title and authors. Duplicates are also those papers that are published by the same authors with a slight change in the title name. In this case, the latest version of the paper was retained.
   - **EC3: Is the paper written in English?** Papers that are present in different language were not considered as it is not possible to understand them.

4.4 Data Analysis

Based on the information obtained by reading the papers, we assigned a tag to relevant phrases and added it to a spreadsheet to keep a record. Each tag assigned to a phrase was based on the context. For example, a phrase about the definition of WIP was assigned a tag called *WIP definition* or a phrase presenting a performance metric to identify WIP was assigned a *WIP metric* tag. We categorized the tags into different categories based on the similarity. For example, all the tags that define, describe, or provide any examples related to handoff waste were included in the handoff subcategory or the tags that define, describe, provide examples, or any metrics used to identify transportation waste were included in the overview subcategory of transportation waste. Figure 5 shows the categories derived from the analysis. Furthermore, when the seven wastes appeared to be the main categories, we modified the existing tags and categories to summarize data further. For example, a tag stating that in a business process, transportation waste expresses itself as handoffs helped us consider the handoff category as a subcategory of transportation waste. These steps were performed iteratively (see Figure 4) until the different categories and subcategories began to stabilize. After theoretical saturation, a final review was performed to verify that the tags associated with a category were correct. This process resulted in 1096 tags categorized into 23 subcategories and 9 main categories.

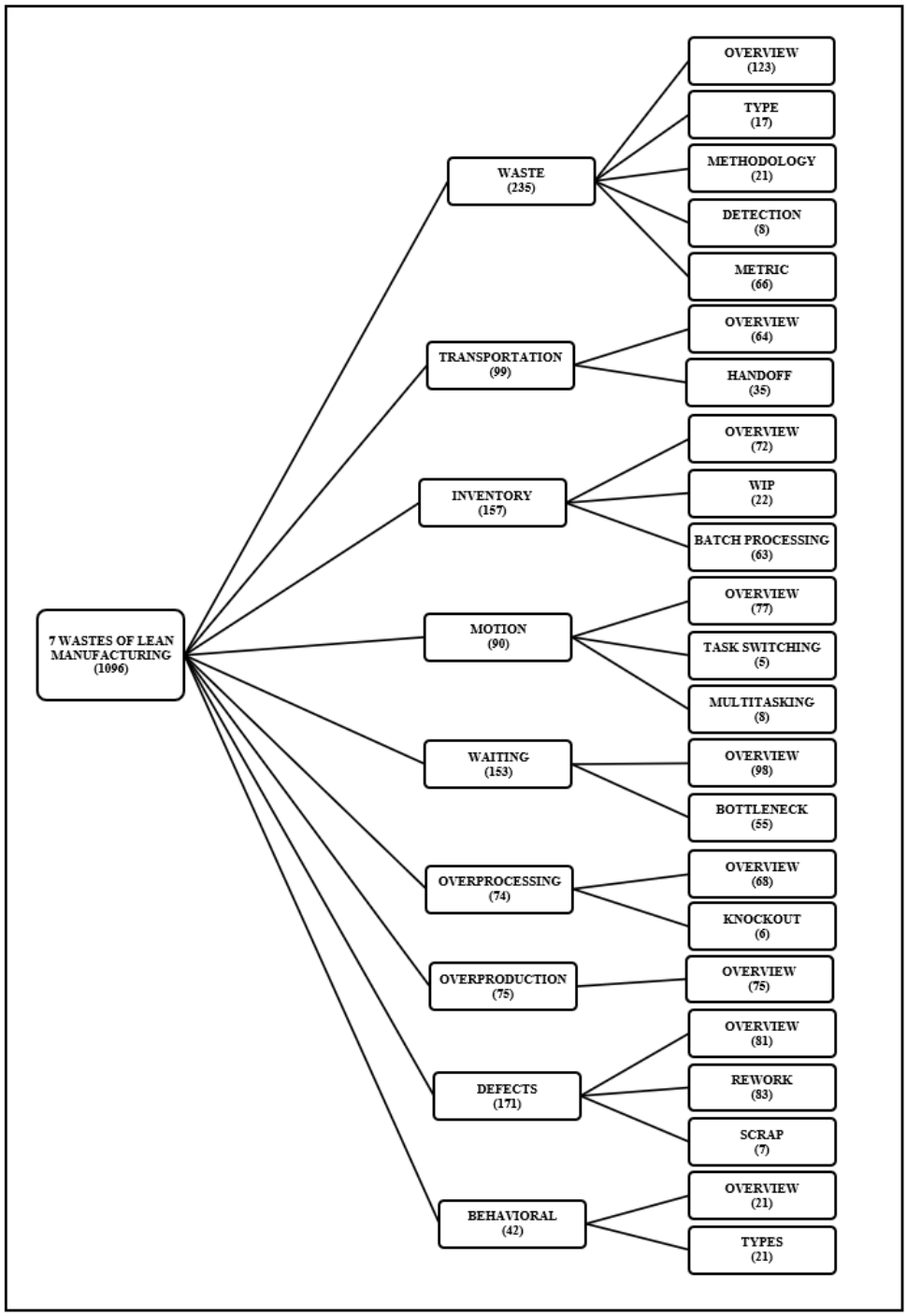

Figure 5: Main Waste Categories

4.5 Framework Elicitation

For the development of the framework, the results of the content analysis were taken as input. Specifically, for each waste, the tags capturing the *definition*, *description*, and *example* were considered. If available, the tags of *cause* and *metric* were also considered. These were examined in detail to gain a deeper understanding of the ways each waste can manifest itself in a business process.

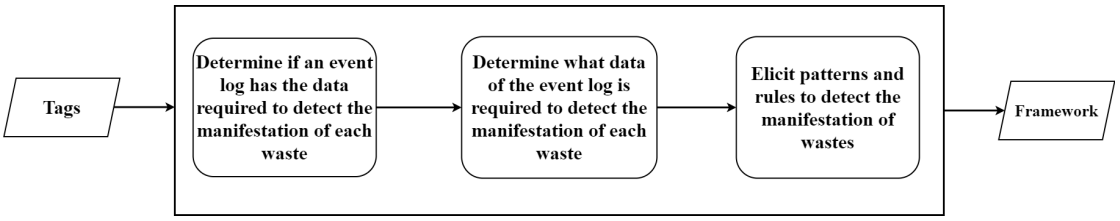

Figure 6: Framework Elicitation Process

Figure 6 shows the process of eliciting the framework. The next steps include determining if an event log has the data required to detect the manifestation of each waste, identifying what data of event log is required to detect these manifestations, eliciting the different patterns and rules based on the data present in the event log, and finally including it in the framework.

Step 1: Determine if an event log has the data required to detect the manifestation of each waste in a business process

Following the examination of the tags, the next step was to determine what data would be required to detect the manifestation of each waste.

For instance, one of the wastes concerns unnecessary transportation of materials. In other words, if an item in a manufacturing process has to be transported from one physical location to another for further processing, moving the item is considered a waste as it adds no value to the end result. Therefore, to detect such a waste, the event log must hold data on each item and its movement through the manufacturing process.

Another example would be the waiting waste, which is defined as the delay between the end of a process or activity and the start of the next process or activity [14]. Therefore, to detect this waste, the event log must hold the start and the end timestamps of each activity.

Step 2: Determine what data of the event log is required to detect the manifestation of each waste

In the second step, the data required for identifying a waste is compared with the data commonly captured in an event log. Event logs hold data that relate to the execution of

a case in a process. As such, data that relates to activities outside of the process is not held. For instance, in the case of transportation, event logs do not capture the data on items' movement. An event log has information on the start and end of an activity, but not on the events taking place outside of the process, such as if an item was transported, and if so, from where to where and how long it took. The wastes that require data not readily available in an event log are out of the scope of this thesis, and therefore, not considered.

Step 3: Elicit how, using the data, the manifestation of each waste can be detected from an event log

From step 2, it is known what manifestations of wastes are possible to detect from an event log. Therefore, for each waste, its different manifestations in a process are considered in light of what can be detected from an event log. In this step, a waste can be manifested in multiple ways. For instance, handoff waste can be expressed in 10 ways. From the analysis of tags, handoff occurs when a case is handed from one resource to another. However, in a process, the resource can be internal and external, intra- or inter-department, or human or system. Therefore, a handoff is manifested as a change between any of the above-mentioned resources. For each of these manifestations, a formula was elicited for detecting it from an event log.

This process of analyzing the tags, determining what data is required, comparing that with data in an event log, and elicitation of how to detect a manifestation of a waste from an event log was conducted for each waste and its specific manifestations. The result of all these steps is a framework that outlines the data-driven rules to identify the different types of wastes from event logs.

5 Results

This section presents the results of the blended content analysis method. The results describe the tags, subcategories, and main categories obtained from the tagging and categorizing process. A total of 9 main categories were identified from tagging the information present in 187 papers: waste, transportation, inventory, motion, waiting, overprocessing, overproduction, defects, and behavioral. These categories were further divided into subcategories and presented by following a drill-down approach. Each category contains an *Overview* subcategory that covers aspects like the definition of the category, a detailed description about a category, different examples, causes of a particular waste, metrics that can be used to calculate the impact of this waste, and any relation to other wastes or categories. This is then followed by presenting the different subcategories that mainly cover the different ways a specific category or waste expresses itself. The 9 main categories are further discussed in detail in the following subsections. The full list of tags and the associated content is not included in the thesis due to space limitations. Therefore, the link is provided to view the full list of tags as a Google Sheet table [1].

5.1 Waste

The *Waste* category includes information related to overview of the seven wastes, different methodologies used to identify the seven types of waste, detection methods to identify the seven wastes, and the metrics associated with all the wastes combined. This category aims to touch upon the basic information about the seven wastes of lean management. A total of 235 tags are included in the waste category. Based on the similarity of tags, the *Waste* category is further divided into 5 subcategories: overview, type, methodology, detection, and metric. Table 1 shows the tag frequency and the sample references associated with each subcategory.

Table 1: Waste Category Tags

Main Category	Subcategory	Tag Frequency	Sample References
Waste	Overview	123	[1, 4, 12, 13, 25, 29, 35–37]
	Type	17	[13]
	Methodology	21	[9, 11, 25, 38]
	Detection	8	[9, 39]
	Metric	66	[24, 40]

The *Overview* subcategory includes 123 tags related to the definition of the waste, a detailed description of the seven wastes combined, different examples of wastes, the

[1] `https://bit.ly/3ybhkas`

possible causes of these wastes, and the relation of the seven wastes with each other. Table 2 shows the tag frequency and sample references associated with the overview subcategory.

Table 2: Waste Overview Subcategory Tags

Category	Tags	Tag Frequency	Sample References
Overview	Definition	75	[1, 4, 11, 13, 25, 35, 36]
	Description	15	[12, 37, 41]
	Example	27	[12, 37, 41, 42]
	Cause	4	[35]
	Waste Linkage	2	[29]

The *Definition* tag covers the definition of waste. Many authors [1, 13, 25, 35] define wastes as anything that consumes resources but does not provide any value to the customer. In terms of software or product development, waste is defined as anything that does not make it to the release [4, 11, 36]. The *Description* tag further describes the waste in detail. Hicks et al. [12] mapped the concepts of flow excess, flow demand, failure demand, and flawed flow to overproduction, waiting, overprocessing, and defects. The different examples of wastes as mentioned in different case studies are associated with the *Example* tag. Falk et al. [42] considered manual data entry of student's data in an admission process as an activity that consumes a lot of time and resources. Many authors [12, 37, 41] described and gave various examples of wastes. A few of the tags associated with the causes of wastes are tagged as *Cause*. For example, Poppendieck et al. [35] considered that large batches of partially done work was the root cause of many of the wastes in software development. Lastly, the relation of the seven wastes with each other is tagged as *Waste Linkage*. Rawabdeh et al. [29] considered all wastes to be interdependent. For example, overproduction was considered the most serious waste leading to a rise in inventory waste [29].

The *Type* subcategory includes 17 tags associated with the different types of waste. Dumas et al. [1] categorized the seven types of wastes into move, hold, and overdo waste. Thurer et al. [13] described two generic types of wastes: obvious and buffer waste. Obvious waste is any waste that can be reduced without creating additional wastes [13]. In contrast, buffer waste is described as any waste that cannot be reduced without creating additional waste [13].

The *Methodology* subcategory includes 21 tags. This category discusses different methodologies like Lean, 5S, Kaizen, and DMAIC [9, 11, 25, 38]. Lean is an approach that focuses on improving quality and eliminating non-value-adding activity [38]. Kaizen provides a framework for allowing workers to continuously improve a process by motivating the employees to come up with improvement ideas [39]. The Motorola company

identified a pattern of improvement that can be divided into five phases of problem-solving: define, measure, analyze, improve, and control [25].

The *Detection* subcategory includes the various detection methods like VSM and WID [9, 39]. VSM represents the production flows and represents the entire value chain [9]. VSM is one of the lean practicies that represents the entire business process value chain [9]. Both VSM and WID enables the visualization between the current and future state [18].

Lastly, 66 tags formed the *Metric* subcategory. This subcategory included the different waste-related metrics such as cycle time, queue time, trace length, and case or activity level metrics [24, 40].

To sum up, the *Waste* category provides a comprehensive view to start analyzing the seven types of wastes in detail.

5.2 Transportation

The *Transportation* category consists of 99 tags that provide a comprehensive overview of transportation waste and the type of transportation waste, i.e., Handoff. This category is further divided into two main subcategories: *Transportation Overview* and *Handoff* (see Figure 5).

The tags related to the definition of transportation waste, a detailed description of transportation waste, different examples of transportation waste, the possible causes of transportation waste, and the relation of transportation waste with other wastes forms the *Transportation Overview* subcategory. Table 3 shows the tag frequency and sample references associated with the *Transportation Overview* subcategory.

Table 3: Transportation Overview Subcategory Tags

Category	**Tags**	**Tag Frequency**	**Sample References**
Transportation Overview	Definition	30	[1, 13, 14, 25, 43]
	Description	11	[1]
	Example	16	[44–46]
	Cause	4	[47–49]
	Waste Linkage	3	[50, 51]

The *Definition* tags presents the definition of transportation waste. The most common definition of transportation waste is the unnecessary movement of materials, products or information [1, 13, 14, 25, 43]. The detailed description of transportation waste is tagged as *Description*. Dumas et al. [1] described transportation waste to be present wherever a sequence flow goes from one lane to another in a pool. The *Example* tags

include the different examples that have been mentioned in different case studies or as a part of explaining the waste. The most common example was that of a manufacturing industry where the production processes are distributed across two facilities, resulting in a truck moving back and forth with the inventory [44–46]. The *Cause* tags present the most common causes of transportation waste, i.e., poor layout, producing a large number of parts, or any kind of transportation of information [47–49]. The relation between transportation and the other wastes is tagged as *Waste linkage*. Transportation waste is closely associated with waiting time [50]. Excessive movement of information or goods results in longer waiting time [51].

In a business process, transportation occurs when there is a handoff of work between the process participants [1]. Thus, handoffs are considered a type in which transportation waste manifests itself. The *Handoff* subcategory gives an overview of handoff waste and the different types of handoffs in a business process. Table 4 shows the tag frequency and sample references associated with the *Handoff* subcategory.

Table 4: Handoff Subcategory Tags

Category	**Sub Category**	**Tag Frequency**	**Sample References**
Handoff	Handoff Overview	22	[1, 4, 20, 52, 53]
	Handoff Type	13	[20, 22, 52]

The *Handoff Overview* subcategory contains the tags related to the definition of handoffs, the further description of handoff waste, different examples of handoff, different metrics that can be used to calculate the impact of handoffs, and the relation of handoffs with other wastes. As there were only 22 tags associated with the handoff overview category, a collective information is presented briefly. Ramakrishnan et al. [52] defined handoff points as the "points of transfer or exchange of information, knowledge, and/or material in a process between various players involved in that process". A handoff occurs whenever there is a transfer of work or control from one event to another [20]. Aalst et al. [53] described handoffs as the transfer of work from A to B in the same case, when the activity executed by A is immediately followed by activity performed by B. It was further suggested that a handoff could be seen as a relation of the similarity or difference between two events [20]. The similarity and difference metric shows the proportion of attributes that are changing and the proportion that does not change [20]. Similar to transportation waste, handoff entails waiting and delay [1, 4].

The *Handoff Type* subcategory includes 13 tags that present the different types of handoffs. Based on the different combinations of event attributes, handoffs can be further categorized into a combination of people, system, actions, internal/external activity handoff [20, 52]. Finally, an instance of an extreme handoff is a ping-pong situation, where a process participant receives back a case that he/she had handed out to someone else before [22].

5.3 Inventory

The *Inventory* category includes 157 tags that present the overview of inventory waste and the 2 types of inventory waste: WIP and Batch Processing. Based on the similarity of the tags, this category was further divided into 3 subcategories namely *Inventory Overview*, *WIP* and *Batch Processing* (see Figure 5).

The *Inventory Overview* subcategory includes 72 tags related to the definition of inventory waste, detailed description of inventory waste, examples of inventory, possible causes of inventory waste, and the relation of inventory waste with the other types of waste. Table 5 shows the tag frequency and sample references associated with the *Inventory Overview* subcategory.

Table 5: Inventory Overview Subcategory Tags

Category	Tags	Tag Frequency	Sample References
Inventory Overview	Definition	29	[9, 29, 54]
	Description	11	[55, 56]
	Example	20	[48, 57]
	Cause	9	[51, 58]
	Waste Linkage	3	[58, 59]

The *Definition* tags present the definition of inventory waste. In a manufacturing industry, inventory waste is unnecessary storage of products or raw materials [29]. Dinis et al. [9] considered any material waiting to be processed or transported as inventory. Rossi et al. [54] define inventory as a build-up of material or information that is not being used. The *Description* tag describes inventory waste in more detail. For example, Chongwatpol et al. [55] considered excess inventory as WIP inventory that produced more than a customer needs. Hines et al. [56] explained how unnecessary inventory hides a variety of problems such as excessive use of space and increase in lead time. The *Examples* tag presents the various examples of inventory waste. For example, a submittal not being sent forward for review leads to accumulation of work-in-process inventory [48, 57]. The possible causes of inventory waste are captured by the *Cause* tag. Singh et al. [51] considered excessive storage and delay as the cause of inventory. Furthermore, Petersen et al. [58] identified that inventory is caused by batching or unsynchronized concurrent tasks. The *Waste Linkage* tag captures information about the relation of inventory waste with other wastes. A high level of inventory causes waiting times [58]. Moreover, AlBaik et al. [59] considered WIP as a direct result of overproduction and waiting.

In a business process, inventory waste shows up as WIP [1]. Thus, WIP is considered a type in which inventory waste manifests itself. Furthermore, batch processing is linked to inventory waste because the cases pile up for processing [22]. Based on this information, inventory waste can be further divided into two types: WIP and Batch

Processing. The *WIP* subcategory includes 22 tags that present an overview of WIP. The *WIP Overview* subcategory comprises the definition of WIP, description of WIP, a few examples of WIP, metrics that can be used to understand WIP in detail, and the relation of WIP with other wastes. Table 6 shows the tag frequency and sample references associated with the *WIP* subcategory.

Table 6: WIP Subcategory Tags

Category	Sub Category	Tag Frequency	Sample References
WIP	WIP Overview	22	[1, 4, 11, 13, 60–64]

WIP is defined as the number of cases that have started but not yet completed [1, 60]. WIP is further described as partially done work in lean software development [4, 11]. A few of the examples of WIP are the number of available but unused information packages, or 1000 raw materials obtained, but only 6000 the raw materials are used to make a garment [61, 62]. The most common metric associated with WIP is longer lead times leading to a delay [13, 63]. Reduction in WIP leads to a reduction in waiting time [64].

The *Batch Processing* subcategory includes 63 tags that present the overview of batch processing, and the 3 different types of batch processing: Simultaneous, Sequential, and Concurrent Batch Processing, Table 7 shows the tag frequency and sample references associated with the *Batch Processing* subcategory.

Table 7: Batch Processing Subcategory Tags

Category	Sub Category	Tag Frequency	Sample References
Batch Processing	Batch Processing Overview	49	[22, 24, 58, 65–69]
	Batch Processing Type	14	[23, 65, 69]

The *Batch Processing* subcategory presents the definition of batch processing, further description of batch processing, a few examples of batch processing, the metrics to measure batch processing, and the relation of batch processing with other wastes. Batch processing is defined as processing a number of cases simultaneously [65]. Swennen et al. [24] defined batch processing as the piled-up activities that are handled simultaneously by the same resource. Thus, batch processing is a collective execution of several instances of different activities. Furthermore, Nakatumba et al. [66] describe a method to identify batch processing by grouping the execution of activities in a chunk and starting the chunk when the elapsed time between the end of an action and the start of the following action exceeds one hour. Pufahl et al. [67] described a batch region to comprise of a set of activities that is executed together or almost immediately after each other. An example of batch processing is in the context of online retailers [68]. When

customers place multiple orders in a short time frame, all these orders are packed and shipped together by the online retailer [68]. The different metrics that can be used to measure batch processing are: the number of times batch processing occurs in an event log, the size of the batch, the number of cases that are included in a batch, the waiting time of activity instances in a batch, and the duration of the batch [24, 69]. Batch processing can be linked to inventory since cases are piled up, waiting since a case needs to wait for an execution of a batch to be completed, and overproduction waste because the outcome of batch processing cannot be picked up together through the next activities [22, 58, 65].

Based on the task and case based orientation of a batch, batch processing is divided into 5 types: Simultaneous batching, Sequential task-based batching, Sequential case-based batching, Concurrent task-based batching, and Concurrent case-based batching [23, 65, 69]. The *Batch Processing Type* subcategory includes all the tags associated with the different types of batch processing. Simultaneous batch processing occurs when instances of an activity are executed by the same resource for different cases at the exact same time [23, 65, 69]. For example, resource R performs a task T for cases 1,2, and 3 simultaneously. Sequential task-based batch processing occurs when the same activity instance is executed sequentially by the same resource in different cases [23]. In sequential case-based batching, a resource performs a set of activity instances sequentially in a particular case, and almost immediately afterwards it performs the same sequence of activity instances [23]. Concurrent task-based batching occurs when there is an overlap in time when a resource executes instances of an activity in distinct cases [23, 65, 69]. On the other hand, concurrent case-base batching occurs when a set of activity instances are executed by the same resource in one case such that there is an overlap in time before proceeding to the next case [23]. There is an overlap in time between the activity instances within and between the cases [23].

5.4 Motion

The *Motion* category includes 90 tags that provide an overview of motion waste and the two types of motion waste: Task Switching and Multitasking. This category is further divided into 3 subcategories namely *Motion Overview*, *Task Switching*, *Multitasking* (see Figure 5).

The *Motion Overview* subcategory includes 77 tags associated with the definition of motion waste, a description of motion waste, few examples of motion, and the relation of motion waste with the other wastes. Table 8 shows the tag frequency and sample references associated with the *Motion Overview* subcategory.

Table 8: Motion Overview Subcategory Tags

Category	Tags	Tag Frequency	Sample References
Motion Overview	Definition	35	[1, 9, 12, 14, 36]
	Description	11	[13, 45]
	Example	29	[70, 71]
	Waste Linkage	2	[29]

The *Definition* tag covers the definition of motion waste. Many authors [1, 9, 12, 14, 36] define motion waste as the unnecessary movement of resources during the execution of a task that does not add any value. The *Description* tag is associated with the information that further describes motion waste. For example, motion can be considered as bending, turning, reaching, and lifting of equipment needed to complete a task [45]. Thurer et al. [13] considered motion as the needless repetitious movements that must be eliminated immediately. The *Example* tag presents the different examples of motion waste. For example, a worker walking few steps to fetch brick at a construction site [70] or when an employee spends a lot of time moving from one data system to another to obtain information required for a task [71]. The *Waste Linkage* tags highlight the relation of motion waste with the other wastes. Motion results in waiting, inventory piling up, and can also lead to more defects [29].

Another form of motion waste that can be seen in digitized processes is task switching which refers to a process participant switching from one application to another during the performance of a task [1]. Thus, motion waste expresses itself in the form of task switching. Task switching is very closely related to multitasking as it describes the same behavior, i.e., performing multiple tasks [4]. Based on this information, the motion category is further divided into *Task switching and Multitasking* subcategories. Table 9 and Table 10 shows the tag frequency covering the overview of both these categories.

Table 9: Task Switching Subcategory Tags

Category	Sub Category	Tag Frequency	References
Task Switching	Task Switching Overview	5	[1, 72, 73]

The *Task Switching* subcategory only has 5 tags that cover basic information about task switching. Skaugset et al. [72] define task switching as the switch between two tasks sometimes done very rapidly. Korkala et al. [73] further described the motion waste as the time taken to re-orient to a particular task after switching between multiples tasks is waste.

Table 10: Multitasking Subcategory Tags

Category	Sub Category	Tag Frequency	Sample References
Multitasking	Multitasking Overview	8	[4, 74, 75]

The *Multitasking* subcategory includes 8 tags that present an overview of multitasking. Skaugset et al. [72] define multitasking as the simultaneous performance of two or more tasks. The parallel execution of tasks is considered to be an extreme form of overlapping [74]. Furthermore, multitasking increases the lead time [75].

5.5 Waiting

The *Waiting* category consists of 153 tags that present the overview of waiting waste and the most common way it expresses itself: Bottleneck. Based on the similarity of tags, this category is divided into 2 subcategories: *Waiting Overview* and *Bottleneck* (see Figure 5).

The *Waiting Overview* subcategory comprises 98 tags that provides a comprehensive overview of the definition of waiting waste, a description of waiting waste, different examples to understand waiting waste, the cause of waiting waste, different metrics to measure waiting waste, and the relation of waiting waste with other types of waste. Table 11 shows the tag frequency and sample references associated with the *Waiting Overview* category.

Table 11: Waiting Overview Subcategory Tags

Category	Tags	Tag Frequency	Sample References
Waiting Overview	Definition	26	[1, 13, 14]
	Description	18	[51, 55, 76]
	Example	44	[59, 77]
	Cause	4	[59, 76]
	Metric	4	[24, 40]
	Waste Linkage	2	[29]

The *Definition* tag defines the waiting waste. Thurer et al. [13] defined waiting as any delay in the actions to achieve process transformation. Waiting can also be defined as any delay between the end of a process or activity and the start of the next process or activity [14]. Based on the collective information extracted from the definition tags, it can be derived that the waiting waste is any delay generated from a task waiting for a resource or the resource waiting for a task. The *Description* tag covers the further explanation of waiting waste. Chongwatpol et al. [55] explained waiting waste by capturing periods of inactivity or time lost when WIPs are waiting to be processed. The different

ways in which waiting is visible in software development can be through waiting for data or a decision of a resource [76]. Furthermore, Singh et al. [51] considered wastes to be identified by the long periods of inactivity pertaining to the people, information, or goods. The *Example* tag presents the various examples of waiting waste as covered in different case studies or research. For example, in the manufacturing industry, waiting waste occurs when a worker has to wait for a task that is currently done by someone else or for a material to be delivered [59]. A patient waiting at each step: registration, triage, or placement in examination room contributes to waiting waste [77]. The *Cause* tag includes the possible causes of waiting waste. According to Gautam et al. [76], the task and arrival variation contributes to long waiting times. At times, a centralized decision-making model in an organization leads to waiting waste [59]. Different metrics are tagged as *Metric*. Waiting time in event logs can be calculated as the sum of unused time in each case [24]. Kupiainen et al. [40] used lead time, processing time, and queue times to detect waiting wastes in the testing phase of the software. The relation of waiting waste with other wastes is captured by the *Waste Linkage* tag. According to Rawabdeh et al. [29], waiting waste is a result of all types of waste as it leads to an increase in the manufacturing lead time.

The longest waiting time in a process are bottlenecks [60]. Thus, waiting time expresses itself as a bottleneck. The *Bottleneck* subcategory comprises 55 tags that cover the overview of bottlenecks. The tag frequency and the sample references associated with the *Bottleneck Overview* subcategory is shown in Table 12.

Table 12: Bottleneck Subcategory Tags

Category	**Sub Category**	**Tag Frequency**	**Sample References**
Bottleneck	Bottleneck Overview	55	[24, 26, 40, 60, 64, 78–80]

The *Bottleneck Overview* subcategory includes tags associated with the definition of bottleneck, a few examples of bottlenecks, information about the detection of bottlenecks and how to identify a bottleneck machine, metrics used to identify the bottleneck, and the relation of bottleneck with other wastes. A bottleneck is an activity or a machine whose performance affects other activities or the overall system [24, 78]. Roser et al. [81] considered bottleneck as the machine with the longest active period. A bottleneck can be caused due to resources with the highest load [60]. Bottlenecks can be detected by identifying tasks or activities with high waiting time [26]. Roser et al. [79] detect bottlenecks by measuring the waiting time in front of a machine or calculating the percentage of the time a machine is active. The different metrics that can be used to identify bottleneck are throughput, queue time, waiting time, active and inactive time [24, 40, 79, 80]. Lastly, the occurrences of bottlenecks can cause a delay in a process [64].

5.6 Overprocessing

The *Overprocessing* category has 74 tags that present the overview of overprocessing waste and a type of overprocessing waste: knockout. This category includes 2 subcategories: *Overprocessing Overview* and *Knockout* (see Figure 5).

The tags related to the definition of overprocessing waste, a detailed description of overprocessing waste, a few examples of overprocessing waste, the factors that cause overprocessing waste, and the relation of overprocessing waste with all the other wastes forms the *Overprocessing Overview* subcategory. A total of 68 tags provides an overview of overprocessing waste (see Table 13)

Table 13: Overprocessing Overview Subcategory Tags

Category	Tags	Tag Frequency	Sample References
Overprocessing Overview	Definition	29	[1, 13, 14, 21]
	Description	14	[55, 82]
	Example	21	[1, 59, 83]
	Cause	3	[49, 76]
	Waste Linkage	1	[50]

The *Definition* tag defines overprocessing waste. Dumas et al. [1] defines overprocessing as the "work that is performed unnecessarily given the outcome of a process instance". Overall, overprocessing is doing unnecessary processing on a task [13, 14, 21]. The *Description* tag describes overprocessing waste in detail. Chongwatpol et al. [55] described overprocessing as WIP that is wrongly processed at a wrong workstation at the wrong time. Substituting a material with a more expensive one to deliver better performance, the use of high-quality equipment where a piece of normal equipment can get the job done, or the execution of simple tasks by an overqualified worker is overprocessing [82]. The different examples of overprocessing are tagged as *Example*. A technician taking a lot of time to measure vehicle emissions, unnecessary record keeping, data collection, or analysts defining requirements that are not required are a few examples of overprocessing [1, 59, 83]. The possible causes of overprocessing waste are tagged as *Cause*. Overprocessing can be caused by solving the same problem repeatedly or using more resources than needed [49, 76]. The relation of overprocessing waste with other wastes is captured by the *Waste Linkage* tag. In a lean supply chain management, eliminating overprocessing waste leads to the elimination of waiting and transportation waste [50]. Thus, overprocessing waste can be associated with waiting and transportation waste.

The *Knockout* subcategory provides a brief overview of the knockout checks that leads to unnecessary processing of tasks. A knockout check is defined as an activity that classifies a case as accepted or rejected, such that if the case is rejected, the effort spent

on all the previous activities becomes overprocessing waste [1, 21]. Table 14 shows the tag frequency and sample references associated with the *Knockout* subcategory.

Table 14: Knockout Subcategory Tags

Category	Sub Category	Tag Frequency	Sample References
Knockout	Knockout Overview	6	[1, 21, 84]

A typical example of a knockout check is an application-to-approval process, where an application goes through several eligibility checks [21]. The check may lead to an application being rejected, and thus all the steps performed before that are considered as waste [1]. The different metrics associated with a knockout check are average processing time, a failure probability, a reject probability, and a set-up time ratio [84].

5.7 Overproduction

The *Overproduction* category consists of 75 tags that present the overview of overproduction waste. This category has only one subcategory: *Overproduction Overview*. The *Overproduction Overview* covers the following aspects: definition of the overproduction waste, information that further describes overproduction waste, examples of overproduction waste, the different metrics related to overproduction waste, and the relation of overproduction waste with other wastes. The tag frequencies and sample references are depicted in Table 15.

Table 15: Overproduction Overview Subcategory Tags

Category	Tags	Tag Frequency	Sample References
Overproduction Overview	Definition	36	[9, 43, 46, 48, 85, 86]
	Description	8	[4]
	Example	22	[1, 87]
	Metric	2	[29, 55]
	Waste Linkage	7	[29, 56, 71]

The *Definition* tag covers the definition of overproduction waste. A typical definition of overproduction waste is producing more, sooner, or faster than is required by the following process [9, 43, 46, 48, 85, 86]. The further description of the overproduction waste is captured by the *Description* tag. In software development, overproduction waste is described as adding extra features that are not necessary for the user's current needs [4]. The examples of overproduction waste are tagged as *Example*. In manufacturing, overproduction waste typically refers to the excess quantity of products [87]. Another example of overproduction waste is that the customer obtains a quote but does not submit a purchase order. The reject quotes are an example of overproduction waste

[1]. The *Metric* tag covers the few metrics associated with overproduction waste. Lead time, number of produced units and demand (number of needed units) are a few of the metrics that can be used to understand the impact of overproduction waste [29, 55]. Finally, the relation of overproduction waste with other wastes is captured by the *Waste Linkage* tag. Producing more items than needed leads to excessive WIP or inventories [29, 56]. The management of this excessive inventory further involves unnecessary transportation and motion [71]. Therefore, overproduction waste is considered the most serious waste as it impacts a vast majority of the other waste.

5.8 Defects

The *Defects* category consists of 171 tags that cover the overview of defects waste and the types of defect waste. Based on the analysis, two subcategories are uncovered: Rework and Scrap. Thus, this category is divided into 3 subcategories: *Defects Overview*, *Rework*, and *Scrap* as shown in fig. 5.

The *Defects Overview* subcategory presents the definition and further description of defect waste, a few examples of defect waste, the possible causes of it, and the relation of defect waste with other types of waste. The tag frequency and associated references are presented in Table 16.

Table 16: Defect Overview Subcategory Tags

Category	Tags	Tag Frequency	Sample References
Defect Overview	Definition	30	[1, 13, 14, 37, 88]
	Description	16	[22, 89]
	Example	31	[1, 48, 90, 91]
	Cause	1	[59]
	Waste Linkage	3	[29]

The definition of defect waste is tagged as *Definition*. Dumas et al. [1] defined defect waste as the work performed to correct, repair, or compensate for a defect in a process. Defects refer to as any activity that leads to rework or scrap [13, 14, 37, 88]. The *Description* tag further describes defects. Delias et al. [22] considered the loops formed by the activities that are repeated during a single case in an event log as a defect. Jones et al. [89] further described defects as product and service defects. Product defects are defined as the defects in goods produced, while service defects are the problems given to a customer that are not directly related to the production of goods. A few of the examples of defect wastes are tagged as *Example*. An example of a defect waste is when a request is sent back by the approver to the requestor for revision due to missing data [1]. Data entry errors, iterations performed to correct something, resubmitting a request are a few of the examples of defect waste [48, 90, 91]. The *Cause* tag describes the possible

causes of defect waste. Inappropriate assumptions can lead to defects [59]. The relation of defect waste and all the other wastes are tagged as *Waste Linkage*. Defect waste is very closely related to transportation waste as it causes wasteful transportation activities due to rework ad scrap [29]. In addition to this, defect waste leads to overproduction, inventory, motion, and waiting waste [29]. Thus, it can be concluded that defect waste affects all the other types of waste.

Based on the different definitions, defects encompasses rework and also leads to scrap [1, 13, 14]. Therefore, defects waste expresses itself in the form of rework and scrap. The *Rework* subcategory provides an overview of rework and the different types of rework. Table 17 shows the tag frequency and the sample references associated with this category.

Table 17: Rework Subcategory Tags

Category	**Sub Category**	**Tag Frequency**	**Sample References**
Rework	Rework Overview	69	[1, 24, 92–95]
	Rework Type	14	[24, 28, 30, 69, 96]

The *Rework Overview* subcategory presents the definition and detailed description of rework, a few examples of rework, different causes of rework, and metrics to measure rework. Rework is defined as doing an activity at least once due to a defect or non-conformance [92]. Cass et al. [93] defined rework as an activity of reconsidering and modifying an earlier decision. In case of an event log, rework is performing an activity again in the same case [1]. Rework is further described as refactoring in which developers perform the transformation in small steps in a software project [94]. Fairley et al. [94] considered rework to be evolutionary, retrospective, and corrective. Evolutionary rework adds value to a product by modifying the current version [94]. In retrospective rework, developers modify the previous version because they did not implement it in such a way that the future capabilities are served [94]. Corrective rework occurs when the developer corrects the defects [94]. The possible causes of rework are task deferral, exceptions, the information needed for performing the task is not available, defects [28, 93, 95]. The metrics related to the self-loops and repetition performed by the same or different resource can be used to measure rework [24].

The *Rework Type* includes tags that present the different types of rework. In an event log, rework can be identified by identifying *self-loop/directly-follows loop* and *back-forth loop/eventually-follows loop* [28, 30, 69, 96]. A self-loop is formed by all the activity instances of the same activity type executed more than once immediately after each other [96]. Swennen et al. [24] further classified self-loop into 2 subtypes: *Repeat self-loop* and *Redo self-loop*. Repeat self-loop occurs when the same resource R executes the same activity immediately after each other within the same case [24]. A Redo self-loop occurs when the same activity is executed immediately after each other by two

different resources R1 and R2, within the same case [24]. An eventually-follows loop is a pattern where the work is returned to the same resource which started it [30]. The eventually-follows loop is further classified into *Redo repetition* and *Repeat repetition* [24]. Redo repetition occurs when two different resources perform an activity within the same case but not immediately after each other [24]. Repeat repetition occurs when the same resource executes an activity within the same case but not immediately after each other [24].

Since defects lead to scrap, the *Scrap* category captures a brief overview of this aspect of defect waste. The tag frequency and sample references of this category are presented in Table 18.

Table 18: Scrap Subcategory Tags

Category	Sub Category	Tag Frequency	Sample References
Scrap	Scrap Overview	7	[56, 97]

Scrap is defined as the production of defects that are not detected [56, 97]. The presence of defects leads to the creation of scrap.

5.9 Behavioral

The *Behavioral* category presents the waste that involves human behavior that adds no value. While the seven types of wastes are the classical wastes, behavioral waste adds a subjective human component. Thus, the seven wastes are the system variables and considered to be different from human variables like unused human potential or wishful thinking [13]. The 8th waste is considered to be the waste of human potential. For example, unclear communication between humans may not be a waste on its own but rather lead to waiting, motion, or overproduction [11]. This category is further divided into *Behavioral Overview* and *Behavioral Waste Type* subcategory (see Table 19).

Table 19: Behavioral Category Tags

Main Category	Subcategory	Tag Frequency	Sample References
Behavioral	Behavioral Overview	21	[4, 11, 13, 98, 99]
	Behavioral Waste Type	21	[11, 54]

The *Behavioral Overview* subcategory includes a few definitions and a variety of examples. Behavioral waste is defined as human behaviors that do not add any value [98]. A few examples of behavioral waste are: assigning a highly specialized task to an untrained worker, not using people's skill to their full potential, and lack of communication that leads to additional waste [4, 11, 13, 99]. Thus, behavioral waste also plays

a role in creating additional waste in a business process such as waiting, motion, and overproduction [11].

The different types of behavioral waste are included in the *Behavioral Waste Type* subcategory. Unmet human potential/unused employee creativity and wishful thinking are the different types of behavioral waste [11, 54]. Unmet human potential is the waste of not using people's skills to their full potential [11]. Wishful thinking is defined as making decisions without taking into consideration all the parameters [11]. Unused employee creativity is not utilizing an employee's talent for the right task.

Since the *Behavioural* category captures the human behavior, it is not possible to detect it from event logs. Thus, it is not included in the framework presented in section 6.

6 Waste Identification Framework

This section presents the data-driven rules to identify wastes from event logs. The results obtained from the blended content analysis method revealed the different types of wastes, the different identification techniques discussed in various case studies, and the patterns associated with different wastes that can be identified from event logs. The proposed framework leverages the results obtained from the blended content analysis method to confirm and complement the different patterns associated with wastes in event logs. The subsequent sections present the different patterns and steps to identify the wastes from event logs. Behavioral waste is excluded from further analysis as this is highly dependent on human aspects and cannot be detected from event logs.

6.1 Preliminaries

Most enterprise systems captures the execution of events [1]. An event log is a collection of timestamped event records [1]. Each record is a touchpoint of the execution of a task. Each event has a unique event ID and refers to a case, a timestamp, a resource, and an activity [1]. Figure 7 shows an excerpt of an event log.

Case ID	Event ID	Timestamp	Activity	Resource
1	Ch-4680555556-1	2012-07-30 11:14	Check stock availability	SYS1
1	Re-5972222222-1	2012-07-30 14:20	Retrieve product from warehouse	Rick
1	Co-6319444444-1	2012-07-30 15:10	Confirm order	Chuck
1	Ge-6402777778-1	2012-07-30 15:22	Get shipping address	SYS2
1	Em-6555555556-1	2012-07-30 15:44	Emit invoice	SYS2
1	Re-4180555556-1	2012-08-04 10:02	Receive payment	SYS2
1	Sh-4659722222-1	2012-08-05 11:11	Ship product	Susi
1	Ar-3833333333-1	2012-08-06 09:12	Archive order	DMS
2	Ch-4055555556-2	2012-08-01 09:44	Check stock availability	SYS1
2	Ch-4208333333-2	2012-08-01 10:06	Check materials availability	SYS1
2	Re-4666666667-2	2012-08-01 11:12	Request raw materials	Ringo
2	Ob-3263888889-2	2012-08-03 07:50	Obtain raw materials	Olaf
2	Ma-6131944444-2	2012-08-04 14:43	Manufacture product	SYS1
2	Co-6187615741-2	2012-08-04 14:51	Confirm order	Conny
2	Em-6388888889-2	2012-08-04 15:20	Emit invoice	SYS2
2	Ge-6439814815-2	2012-08-04 15:27	Get shipping address	SYS2
2	Sh-7277777778-2	2012-08-04 17:28	Ship product	Sara
2	Re-3611111111-2	2012-08-07 08:40	Receive payment	SYS2
2	Ar-3680555556-2	2012-08-07 08:50	Archive order	DMS

Figure 7: Excerpt of an event log of an order-to-cash process [1]

As a minimum requirement, each event should have a case identifier, activity, and end timestamp to perform process mining [1]. In practice, an event log can also have additional attributes like resource who performed a task, cost, or domain-specific data such

as the role of a resource. Table 20 presents the minimum attribute requirements for identifying different manifestations of waste.

Table 20: Required Attributes for Waste Identification

Waste Type	Case ID	Activity	End Timestamp	Start Timestamp	Resource
Handoff	✓	✓	✓	✓	✓
Ping-Pong	✓	✓	✓	✓	✓
WIP	✓	✓	✓	✓	
Batch Processing	✓	✓	✓	✓	✓
Task Switching	✓	✓	✓	✓	✓
Multitasking	✓	✓	✓	✓	✓
Waiting	✓	✓	✓	✓	
Bottleneck	✓	✓	✓	✓	✓
Overprocessing	✓	✓	✓	✓	
Overproduction	✓	✓	✓	✓	
Rework	✓	✓	✓	✓	✓

The different notations used in the next sections are: case C, activity A, end timestamp ET, start timestamp ST, a set of activities A_s, a set of cases C_s, resource R and role RO.

6.2 Transportation

This section presents the different patterns to identify transportation waste from event logs. Transportation waste is the unnecessary movement of materials, products, or information [1, 13, 14, 25, 43]. As mentioned in section 5.2, transportation waste shows up in the form of handoff and ping-pong in a business process. A handoff occurs whenever there is a transfer of work or control from one event to another [20]. Ping-pong is an extreme kind of handoff where a resource receives back a case that he/she passed it on to someone else [22]. In addition to the minimum required attributes, i.e., case id, activity, end timestamp, and start timestamp, the resource information is also needed to detect both these manifestations of transportation waste from event logs.

6.2.1 Handoff and Ping-Pong Preliminaries

Consider an event log L, such that each case C_n contains a set of events E_s such that $E_s = e_1, e_2, e_3 \dots e_i$, where i and n is a unique event and case identifier respectively. To identify handoff or ping-pong, each event e_i has a few required attributes namely an activity A, a start timestamp ST, end timestamp ET, and a resource R such that:

$$attr(A_i \wedge R_i) \in e_i \tag{1}$$

For each transition between events $e_i \rightarrow e_j$, where i, j are the identifiers, a transition between its attributes also occurs.

$$attr(A_i \wedge R_i) \rightarrow attr(A_j \wedge R_j) \tag{2}$$

For any kind of handoff or ping-pong (an extreme kind of handoff) to occur between two consecutive activities, the following conditions should hold:

1. The resources performing the two consecutive activities should be different, i.e. $R_i \neq R_j$
2. The two consecutive activities should be different from each other, i.e. $A_i \neq A_j$
3. The previous activity should be completed before the next activity, i.e. $ET_i \leq ST_j$

6.2.2 Handoff Identification

Pentland et al. [20] identified different handoffs from event logs depending on the change in attributes between events. This includes a handoff from one actor to another, one system to another, one action to another, or the combination of all these attributes changes [20]. A handoff is also classified as an internal or external activity handoff based on whether the activity is internal or external to an organization [52]. The transition between only two actions can also be considered as a handoff [20]. However, within the scope of an event log, every activity transition will be regarded as a handoff. Even though it will capture the sequence of actions where only one resource engages with it, it does not serve the purpose of identifying handoff waste in particular.

A typical handoff can be identified by discovering a switch between the resources for two consecutive activities such that $A_i \neq A_j$, $R_i \neq R_j$, and $ET_i \leq ST_j$.

$$attr(A_i, R_i) \rightarrow attr(A_j, R_j) \tag{3}$$

For different types of handoff, consider that each event e_i has a few additional attributes, namely an activity A, a start timestamp ST, end timestamp ET a human resource HR, a system resource SR, and Role RO such that:

$$attr(A_i \wedge (HR_i \vee SR_i) \vee RO_i) \in e_i \tag{4}$$

For each transition between events $e_i \rightarrow e_j$, where i, j are the identifiers, a transition between its attributes also occurs.

$$attr(A_i \wedge (HR_i \vee SR_i) \vee RO_i) \rightarrow attr(A_j \wedge ((HR_j \vee SR_j) \vee RO_j) \tag{5}$$

For each transition between activities, $A_i \rightarrow A_j$ of events e_i and e_j, a transition between the resources and roles occurs.

$$attr((HR_i \vee SR_i) \vee RO_i) \rightarrow attr((HR_j \vee SR_j) \vee RO_j) \tag{6}$$

Thus, if an event log has attributes related to the type of resources and roles, one can extract more information to identify different kinds of handoffs in a process. Based on the combination of different types of resources and considering whether the resources can be a part of an internal or external organization, handoffs can be of 10 main types: human to human, system to system, system to human, human to system, human(system) to human(system), internal human to external human, internal human to an external system, internal system to external system, internal system to external human, and internal human(system) to external human(system) (refer Table 21).

A *human-human* handoff is a switch between two human resources for two consecutive activities [20]. To dive deeper, one can also consider a human-human handoff with the same or different roles. For example, when resource A, a clerk, handovers a task to resource B, an analyst is considered a human to human handoff having different roles.

$$\begin{aligned} attr(A_i, HR_i, RO_i) \rightarrow attr(A_j, HR_j, RO_j) \\ such\,that\, A_i \neq A_j\,,\; HR_i \neq HR_j\,,\; RO_i \neq RO_j \end{aligned} \tag{7}$$

Table 21: Types of Handoff

Types of Handoff	**Pattern**
Human to Human	$attr(A_i, HR_i) \rightarrow attr(A_j, HR_j)$
System to System	$attr(A_i, SR_i) \rightarrow attr(A_j, SR_j)$
System to Human	$attr(A_i, SR_i) \rightarrow attr(A_j, HR_j)$
Human to System	$attr(A_i, HR_i) \rightarrow attr(A_j, SR_j)$
Human(system) to Human(system)	$attr(A_i, HR_i, SR_i) \rightarrow attr(A_j, HR_j, SR_j)$
Internal Human to External Human	$attr(A_i, HR_i) \rightarrow attr[(A_j, HR_j)]_{ex}$
Internal Human to External System	$attr(A_i, HR_i) \rightarrow attr[(A_j, SR_j)]_{ex}$
Internal System to External Human	$attr(A_i, SR_i) \rightarrow attr[(A_j, HR_j)]_{ex}$
Internal System to External System	$attr(A_i, SR_i) \rightarrow attr[(A_j, SR_j)]_{ex}$
Internal Human(system) to External Human(system)	$attr(A_i, HR_i, SR_i) \rightarrow attr[(A_j, HR_j, SR_j)]_{ex}$

On the other hand, a resource A, a clerk, handovers a task to resource B, another clerk is considered a human to human handover having the same role.

$$\begin{aligned} attr(A_i, HR_i, RO_i) \rightarrow attr(A_j, HR_j, RO_j) \\ such\,that\, A_i \neq A_j\,,\; HR_i \neq HR_j\,,\; RO_i = RO_j \end{aligned} \tag{8}$$

A *system-system* handoff is the switch between two systems for two consecutive activities [20]. Based on the combination of human and system resource attributes, the handoff can also occur between system-human or human-system. A *system-human* handoff occurs when the two consecutive activities are performed by a system and human, respectively. On the other hand, a *human-system* handoff occurs when two consecutive activities are performed by a human resource and a system, respectively. A *Human(system) to Human(system)* handoff denotes the change in both human and system

resources between two consecutive activities. This occurs when a human uses a system to perform an action. Furthermore, based on whether the human or system resource is internal or external to an organization, a handoff can also be of the following types: internal human - external human, internal human - external system, internal system - external system, internal system - external human, and internal human(system) to external human(system). An *Internal Human to External Human* handoff is the switch between human resources within and outside the organization for two consecutive activities. This can also be a handoff with the same or different roles. An *Internal Human to External System* handoff is the switch between a human resource belonging to an organization and a system outside the organization. For example, an employee submits a request to fetch data from a health system. An *Internal System to External Human* handoff is the switch between a system resource present in an organization and a human resource outside the organization. An *Internal System - External System* handoff occurs when there is a switch between two systems present inside and outside an organization, respectively. Lastly, *Internal Human(system) to External Human(system)* handoff occurs when there is a switch between a human using a system within an organization and a human outside an organization using another system. For example, an employee uses a system to complete an activity within an organization and passes it over to another employee in a different organization to perform an activity.

6.2.3 Handoff Metrics

Using the information related to the different handoffs, the different metrics to identify handoffs are identified. The metrics can provide an insight into the impact of handoff waste in a business process. The different metrics are:

1. **Handoff Frequency:** The handoff frequency is the number of times a particular type of handoff between two consecutive activity instances A_i and A_j occurs in an event log L.
2. **Handoff Duration:** The handoff duration is the time taken to switch between two consecutive activity instances A_i and A_j. This is the *handoff waiting time* which is the difference between the start time of A_j and the end time of A_i.
3. **Score:** The score is the result obtained by multiplying frequency and duration.

Thus, a list of all the handoffs and their frequency, duration, and score can be obtained by looking for the specific patterns in an event log. Using these metrics, an analyst can choose to rank the handoffs either by frequency, duration, or score to understand the impact of different types of handoffs in an event log. For example, an analyst can choose to rank the handoffs based on the frequency and consider the top 25% of handoffs as the ones that require immediate attention.

6.2.4 Ping-pong Identification

A ping-pong is identified when a certain resource receives back a case that was handed out to someone else before. In a ping-pong pattern, the resources alternate with one – another [14, 30]. Based on the different patterns observed in an event log, ping-pong can be analyzed at two levels: resource and role level. The conditions as mentioned in section 6.2.1 hold for ping-pong as well.

Table 22: Types of Ping-Pong

Types of Ping-Pong	**Pattern**
Resource level Ping-Pong	*Step1:* $attr(Ai, Ri) \rightarrow attr(Aj, Rj)$ *Step2:* $attr(Aj, Rj) \rightarrow attr(Ai, Ri)$ *Step3:* $attr(Ai, Ri) \rightarrow attr(Aj, Rj)$
Role level Ping-Pong	*Step1:* $attr(Ai, ROi) \rightarrow attr(Aj, ROj)$ *Step2:* $attr(Aj, ROj) \rightarrow attr(Ai, ROi)$ *Step3:* $attr(Ai, ROi) \rightarrow attr(Aj, ROj)$

For any ping-pong pattern to occur, a minimum of 3 handoffs should take place. A *Resource level Ping-Pong* takes into consideration the resource attribute while the *Role level Ping-Pong* considers the role attribute. The resource level ping-pong can be identified when the following transfer of control occurs: Resource R_i completes activity A_i and hands over the work to resource R_j. This is where the first handoff takes place. The second handoff occurs when resource R_j completes activity A_j and hands over the work to resource R_i. Finally, resource R_i completes activity A_i and hands over the work again to resource R_j. This is where the third handoff takes place.

Similarly, the ping-pong at the role level can be detected when the following transfer of control occurs: role RO_i completes activity A_i and hands over the work to role RO_j. This is where the first handoff takes place. The second handoff occurs when role RO_j completes activity A_j and hands over the work to role RO_i. The third and final handoff occurs when role RO_i completes activity A_i and hands over the work again to role RO_j.

6.2.5 Ping-pong Metrics

Similar to handoffs, the different metrics that can provide deeper insight on the impact of ping-pong wastes are:

1. **Ping-pong Frequency:** The number of times the second and third handoff when combined has occurred between two consecutive activities i.e. the occurrences of the pair of Step2 and Step3 in an event log L.
2. **Ping-pong Duration:** The duration is the sum of the time taken by the second

and the third handoff between two consecutive activities i.e. the sum of the time taken by Step2 and Step3 in an event log L.

3. **Score:** The score is the result obtained by multiplying frequency and duration.

Based on these metrics, an analyst can measure the impact of the resource or role ping-pongs by ranking their frequency, duration, or score in descending order.

6.3 Inventory

This section presents the steps to identify inventory waste from event logs. Inventory waste is the build-up of material or information that is not being used [54]. Inventory waste shows up in the form of WIP and batch processing (refer to section 5.3). WIP is the number of cases that have started but not yet completed [1]. Batch processing refers to processing a number of cases simultaneously [23, 65]. The start and end timestamp is required to detect WIP. For batch processing, resource information is also needed. The subsequent sections present the WIP and batch processing detection method from event logs.

6.3.1 WIP and Batch Processing Preliminaries

Consider an event log L, such that each case C_n contains a set of events E_s such that $E_s = e_1, e_2, e_3 \dots e_i$, where i and n is a unique event and case identifier respectively. The calculation of waiting time is a starting point for identification of WIP. The waiting time between two activities WT_{a_i,a_j} is the difference between the start timestamp of the second activity ST_j and the end timestamp of the first activity ET_i.

$$WT_{a_i,a_j} = ST_j - ET_i \tag{9}$$

To identify batch processing, each event e_i should have a few required attributes, namely an activity A, a start timestamp ST, end timestamp ET, and a resource R.

$$attr(A \wedge R \wedge ST \wedge ET) \in e_i \tag{10}$$

6.3.2 WIP Identification

WIP is defined as the number of cases that have started but not yet ended [1]. This indicates that WIP can also be the number of cases that are waiting to be processed. Based on the level of analysis, WIP can be classified into 2 types: *WIP in a log* and *WIP within activities*.

The *WIP in a log* is the number of cases waiting to be processed at a particular time in an event log. To calculate WIP in a log (IW_l), the waiting time of each case in the log

is calculated first. Waiting time for each case CWT_n is the sum of the time difference between the start time of the next activity and the end time of the previous activity in a case. Let there be p number of activity pairs in a case.

$$CWT_n = \sum_{i=1}^{p} [WT_{a_i,a_j}]_i \tag{11}$$

Thus, a log contains a set of case waiting time CWT_{log} such that:

$$CWT_{log} = CWT_1, CWT_2, CWT_3 ... CWT_n \tag{12}$$

The Freedman-Diaconis rule is used to divide the above waiting time range into bins to identify the WIP at a particular point in time. Algorithm 1 presents the steps to calculate the WIP in a log.

Algorithm 1: WIP in a log

Input: CWT_{log}
Result: WIP IW_l in log L
Steps:

1. Find the minimum waiting time of a case CWTmin in the log
2. Find the maximum waiting time of a case CWTmax in the log
3. Calculate the bin width (bw) using the Freedman-Diaconis rule

$$bw = (2 * IQR)/\sqrt[3]{n} \tag{13}$$

   where:
   n = number of cases in a log L
   IQR = $Q3$-$Q1$ such that $Q1$ and $Q3$ is the median of lower and upper range of CWT_{log} respectively
4. Calculate the number of bins (b) = ($CWTmax$-$CWTmin$)/bw
5. Each bin i has its own start and end timestamp - bst_i and bet_i
6. Calculate the number of cases where for each bin $bst_i <= CWT_n <= bet_i$

For visualization purposes, a WIP over time graph can be plotted where the waiting time is plotted on the x-axis plots and the WIP on the y-axis.

The *WIP within Activities* IW_{act} is the number of cases waiting to be processed between two consecutive activities at a particular time. Algorithm 2 presents the steps to identify WIP between two activities.

Algorithm 2: WIP between two activities

Input: Log L

Result: WIP IW_{act} between activities

Steps:

1. Find the pair of consecutive activity instances $A_i \rightarrow A_j$ present in a log
2. For each activity pair, find the number of cases nc in which the pair occurs
   (a) For each case C in which the activity pair occurs
      i. Calculate the end timestamp ET_i of the first activity A_i in the pair to obtain the start of the waiting time
      ii. Calculate the start timestamp ST_j of the second activity A_j to obtain the end of the waiting time
      iii. Calculate the number of concurrent cases (n) between ET_i and ST_j. The number of concurrent cases is calculated by finding the overlaps in waiting time when compared to all the other cases in which the activity pair occurs.
      iv. For each overlapping case C_o, find the concurrent waiting time TCT_o

$$TCT_o = ET_o - ST_o \tag{14}$$

      v. Calculate the total concurrent waiting time TCT_c. The total concurrent waiting time is the summation of the concurrent waiting time of the remaining cases TCT_o in which the activity pair occurred overlaps with case C.

$$TCT_c = \sum_{i=1}^{n-1} [TCT_o]_i \tag{15}$$

      vi. Calculate the score s_c.

$$s_c = n * TCT_c \tag{16}$$

   (b) Calculate the total activity pair score s_{act}.

$$s_{act} = \sum_{i=1}^{nc} [s_c]_i \tag{17}$$

To dig deeper, we can also calculate the *Maximum WIP between Two Activities* to identify the maximum number of concurrent cases and the time it takes between two activities. Algorithm 3 presents the steps to identify maximum WIP in an activity pair.

Algorithm 3: Max WIP between two activities

Input: Log L

Result: Max WIP IW_{act} between two activities

Steps:

1. Find the pair of consecutive activity instances $A_i \rightarrow A_j$ present in a log
2. For each activity pair, find the number of cases nc in which the pair occurs
   (a) For each case C in which the activity pair occurs
      i. Calculate the end timestamp ET_i of the first activity A_i in the pair to obtain the start of the waiting time
      ii. Calculate the start timestamp ST_j of the second activity A_j to obtain the end of the waiting time
      iii. Calculate the max number of concurrent cases (CC_{max}). The max number of concurrent cases are the maximum number of times the waiting time of other cases in which the activity pair occurs overlaps with the waiting time span of case C.
      iv. Calculate the number of occurrences of CC_{max}. This is done to identify the number of times CC_{max} has occurred.
      v. Calculate the start and the end time (ST_{cc} and ET_{cc}) between which the max concurrent cases took place. In case the max concurrent case occurs more than once, find the start and end time for each occurrence.
      vi. Calculate the total maximum concurrent waiting time (TCT_{mc}). In case the max concurrent case occurs more than once, calculate the (TCT_{mc}) for each max occurrence and then calculate the average.

$$TCT_{mc} = ET_{cc} - ST_{cc} \tag{18}$$

      vii. Calculate the score s_c

$$s_c = CC_{max} * TCT_{mc} \tag{19}$$

   (b) Calculate the total activity pair score s_{act}.

$$s_{act} = \sum_{i=1}^{nc} [s_c]_i \tag{20}$$

6.3.3 WIP Metrics

The total activity pair score helps to measure the impact of WIP between activities in a log. The analyst can choose to rank the total activity pair score in descending order to identify the activity pairs with the highest WIP or the maximum concurrent cases.

6.3.4 Batch Processing Identification

Martin et al. [69] defined batch processing as "a type of work organisation in which a resource executes a particular activity on multiple cases simultaneously or concurrently, or intentionally defers activity execution to handle multiple cases sequentially". Batch processing can be performed at the level of a task or a sub-process [23]. Batch processing at the task level considers the same activity performed by the same resource for a group of cases [23]. Batch processing at the case level considers several activities performed by the same resource for a group of cases [23]. Any type of batch should have at least two activities or a set of activities. Based on these dimensions, batch processing can be of 5 types: Simultaneous/Parallel, Sequential task-based, Sequential case-based, Concurrent task-based, and Concurrent case-based [23, 69].

A *simultaneous/parallel* batch processing occurs when the same resource performs one or more activities for a set of cases [23]. A simultaneous/parallel batch occurs when for a set of different cases (C_s), R performs the same activity A in parallel or performs a set of activities A_s = $A_1, A_2, A_3, ...A_k$ in such a way that each activity in the set is performed sequentially in a case and exactly at the same time in different cases. Thus, a minimum of two activities are considered to be a part of a parallel batch when:

$$\begin{aligned} attr(C_i, R_i, A_i, ST_i, ET_i) \rightarrow attr(C_j, R_j, A_j, ST_j, ET_j) \\ where\, C_i \neq C_j,\ A_i = A_j,\ R_i = R_j,\ ET_i = ET_j\, and\, ST_i = ST_j \end{aligned} \tag{21}$$

Since each simultaneous batch contains a set of same activities performed in parallel, multiple parallel batches executed one after another forms a simultaneous batch of multiple consecutive activities.

A *sequential task-based* batch processing occurs when the same resource performs the same activity almost immediately after each other in different cases [23, 69]. A sequential task-based batch occurs when for a set of different cases (C_s), R performs the same activity A almost immediately after each other. Thus, a minimum of two activities are considered to be a part of a sequential task-based batch when:

$$\begin{aligned} attr(C_i, R_i, A_i, ST_i, ET_i) \rightarrow attr(C_j, R_j, A_j, ST_j, ET_j) \\ where\, C_i \neq C_j,\ A_i = A_j,\ R_i = R_j,\ ST_j \geq ET_i,\ ST_j \leq ET_i + tt,\ ARj \leq ST_{fc} \end{aligned} \tag{22}$$

where tt is the tolerance time, AR_j is the arrival time and ST_{fc} is the start timestamp of the first case in a batch. The case arrival is proxied by the completion time of the prior activity executed in that case and tolerance time is the time that is allowed between consecutive instances in a sequential batch [69].

A *sequential case-based* batch processing occurs when the same resource performs a series of tasks sequentially for a case and then performs the same task sequence for

other cases almost immediately after the previous case [23]. A sequential case-based batch occurs when for a set of different cases (C_s), R performs the same set of activities A_s sequentially in an individual case and almost immediately perform the same task sequence in all the other cases in the set. Thus, activities are considered to be a part of a sequential case-based batch when the following conditions hold:

1. For each case C in set, find the activities executed sequentially by the same resource.

$$\begin{aligned} attr(R_i, A_i, ST_i, ET_i) \rightarrow attr(R_j, A_j, ST_j, ET_j) \\ where\, A_i \neq A_j,\, R_i = R_j,\, ST_i \leq ST_j,\, ST_j \geq ET_i,\, ST_j \leq ET_i + tt \end{aligned} \tag{23}$$

2. The same sequence of activities are performed in next case immediately after being performed in the previous case such that the end timestamp of the last activity in the batch of previous case (ET_{pa}) is less than the start timestamp of the first activity in the batch of the next case (ST_{la}).

A *concurrent task-based* batch processing occurs when the activities performed by the same resource in different cases partially overlap in time. A concurrent task-based batch occurs when for a set of different cases (C_s), R performs the same activity A such that there is an overlap in time. Thus, a minimum of two activities are considered to be a part of a concurrent task-based batch when:

$$\begin{aligned} attr(C_i, R_i, A_i, ST_i, ET_i) \rightarrow attr(C_j, R_j, A_j, ST_j, ET_j) \\ where\, C_i \neq C_j,\, A_i = A_j,\, R_i = R_j,\, ST_j \geq ST_i,\, ST_j < ET_i,\, ET_i \neq ET_j \end{aligned} \tag{24}$$

A *concurrent case-based* batching occurs when the same resource performs the same set of activities in different cases in such a way that there is an overlap in time when executing the activities [23]. A concurrent case-based batch occurs when for a set of different cases (C_s), R performs the same set of activities A_s such that there is an overlap in time. Thus, activities are considered to be a part of a concurrent case-based batch when the following conditions hold:

1. For each case C in set, find the activities executed by the same resource such that there is an overlap in time

$$\begin{aligned} attr(R_i, A_i, ST_i, ET_i) \rightarrow attr(R_j, A_j, ST_j, ET_j) \\ where\, A_i \neq A_j,\, R_i = R_j,\, ST_j \geq ST_i,\, ST_j < ET_i,\, ET_i \neq ET_j \end{aligned} \tag{25}$$

2. The same set of activities are performed in the next case by the same resource such that there is an overlap in time between the activities.

6.3.5 Batch Processing Metrics

To gain a deeper understanding, different metrics that describe the batching behavior can be used to gain insight into the impact of batch processing in a business process [69].

1. **Frequency of batch processing:** The number of times that batch contains two or more activity instances is the frequency of batch processing [69].
2. **Size of batch:** A batch size depicts the summary statistics of the number of activities in each batch [69].
3. **Overlapping time:** The amount of time that the activities in a concurrent batch overlap help to analyze the maximum concurrency [69].

All these metrics provide a statistical view on batch processing. However, when ranked in descending order, an analyst can identify the most frequent, time-consuming, or lengthy batches in a log.

6.4 Motion

This section presents the different patterns to identify motion waste from event logs. Motion is the movement of process participants moving from one place to another during the execution of a process [1]. Motion waste expresses itself in the form of task switching and multitasking in a business process (refer to section 5.4). Task switching is the switch between two tasks, and multitasking is the simultaneous performance of two or more tasks [72]. Both these manifestations require the resource information to be present in an event log.

6.4.1 Task Switching and Multitasking Preliminaries

Consider an event log L, such that each case C_n contains a set of events E_s such that $E_s = e_1, e_2, e_3 \dots e_i$, where i and n is a unique event and case identifier respectively. To identify task switching and multitasking, each event e_i should have a few required attributes namely an activity A, a start timestamp ST, end timestamp ET and a resource R.

$$attr(A \wedge R \wedge ST \wedge ET) \in e_i \tag{26}$$

6.4.2 Task Switching Identification

Task switching is the switch between two tasks [72]. Based on this definition, a task switch in an event log is detected when the resource R starts doing another task without completing the previous one. Thus, the end timestamps of the two tasks can be equal to,

less than, or greater than each other. Based on the level of analysis, task switching can be detected within the same case, within different cases, or in the entire log (irrespective of cases).

Task Switching within the Same Case occurs when a resource R does not complete an activity A_i before starting with another activity A_j of the same case. For each transition between events $e_i \rightarrow e_j$, a task switch within the same case occurs when:

$$\begin{aligned} attr(R_i,\, A_i,\, ST_i,\, ET_i,\, C_i) \rightarrow attr(R_j,\, A_j,\, ST_j,\, ET_j,\, C_j) \\ where\, C_i = C_j,\, A_i \neq A_j,\, ST_j > ST_i,\, R_i = R_j \end{aligned} \tag{27}$$

Task Switching in Different Case occurs when a resource switches between different cases when performing the same or different activities. Thus, for further analysis, task switching in different cases is further divided into two types: *same activity* and *different activity*.

Task switching within different cases for executing the same activity occurs when a resource R does not complete an activity A_i of case C_i before starting with the same activity of another case C_j. For each transition between events $e_i \rightarrow e_j$, a task switch between the same activity within different cases occurs when:

$$\begin{aligned} attr(R_i,\, A_i,\, ST_i,\, ET_i,\, C_i) \rightarrow attr(R_j,\, A_j,\, ST_j,\, ET_j,\, C_j) \\ where\, C_i \neq C_j,\, A_i = A_j,\, ST_j > ST_i,\, R_i = R_j \end{aligned} \tag{28}$$

Task switching within different cases for executing different activities occurs when a resource R does not complete an activity A_i of case C_i before starting with another activity A_j of another case C_j. For each transition between events $e_i \rightarrow e_j$, a task switch between the different activities within different cases occurs when:

$$\begin{aligned} attr(R_i,\, A_i,\, ST_i,\, ET_i,\, C_i) \rightarrow attr(R_j,\, A_j,\, ST_j,\, ET_j,\, C_j) \\ where\, C_i \neq C_j,\, A_i \neq A_j,\, ST_j > ST_i,\, R_i = R_j \end{aligned} \tag{29}$$

Task Switch in a log irrespective of cases occurs when a resource R does not complete an activity A_i before starting another activity A_j in a log. This helps us to obtain an overall view of the task switches performed by a resource. In this case, for each transition between events $e_i \rightarrow e_j$, a task switch occurs when:

$$\begin{aligned} attr(R_i,\, A_i,\, ST_i,\, ET_i) \rightarrow attr(R_j,\, A_j,\, ST_j,\, ET_j) \\ where\, A_i \neq A_j,\, ST_j > ST_i,\, R_i = R_j \end{aligned} \tag{30}$$

6.4.3 Multitasking Identification

The simultaneous performance of two or more tasks is known as multitasking [74]. Based on this definition, multitasking in an event log is detected when a resource R performs more than one task at the same time. Based on the start and end timestamp of the activities, multitasking is further divided into two: *Complete Multitasking* and *Partial Multitasking*. Similar to task switching, multitasking can be detected within the same case, within different cases, or in the entire log (irrespective of the cases).

Complete Multitasking within the Same Case occurs when a resource R starts and completes the execution of different activities at the same time within the same case. For each transition between events $e_i \rightarrow e_j$, multitasking within the same case occurs when:

$$\begin{aligned} &attr(R_i,\, A_i,\, ST_i,\, ET_i,\, C_i) \rightarrow attr(R_j,\, A_j,\, ST_j,\, ET_j,\, C_j) \\ &where\, C_i = C_j,\, A_i \neq A_j,\, ST_j = ST_i,\, ET_i = ET_j,\, R_i = R_j \end{aligned} \tag{31}$$

Complete Multitasking in Different Case occurs when a resource R starts and completes the execution of the same or different activities in different cases at the same time. Therefore, this can be further analyzed at the *same activity* and *different activity* level.

Complete multitasking within different cases for executing the same activity occurs when a resource R starts and completes the same activity A_i belonging to different cases at the same time. For each transition between events $e_i \rightarrow e_j$, multitasking with the same activity within different cases occurs when:

$$\begin{aligned} &attr(R_i,\, A_i,\, ST_i,\, ET_i,\, C_i) \rightarrow attr(R_j,\, A_j,\, ST_j,\, ET_j,\, C_j) \\ &where\, C_i \neq C_j,\, A_i = A_j,\, ST_j = ST_i,\, ET_i = ET_j,\, R_i = R_j \end{aligned} \tag{32}$$

Complete multitasking within different cases for executing different activities occurs when a resource R starts and completes the execution of the different activities A_i and A_j belonging to different cases at the same time. For each transition between events $e_i \rightarrow e_j$, multitasking with different activities within different cases occurs when:

$$\begin{aligned} &attr(R_i,\, A_i,\, ST_i,\, ET_i,\, C_i) \rightarrow attr(R_j,\, A_j,\, ST_j,\, ET_j,\, C_j) \\ &where\, C_i \neq C_j,\, A_i \neq A_j,\, ST_j = ST_i,\, ET_i = ET_j,\, R_i = R_j \end{aligned} \tag{33}$$

Complete Multitasking in a log irrespective of cases occurs when a resource starts and completes different activities at the same time in a log.

$$\begin{aligned} &attr(R_i,\, A_i,\, ST_i,\, ET_i) \rightarrow attr(R_j,\, A_j,\, ST_j,\, ET_j) \\ &where\, A_i \neq A_j,\, ST_j = ST_i,\, ET_i = ET_j,\, R_i = R_j \end{aligned} \tag{34}$$

Partial Multitasking is an extension to task switching. Partial multitasking occurs when a resource R performs the same or different activities over a specific time span. This is identified by the concurrent execution of two or more activities during a task switch. The time in which two or more activities are executed concurrently by the same resource is known as concurrent time (TC). Thus, when a task switch is identified, the TC can be calculated based on the relation between ET_i and ET_j in equations 27-30.

$$TC = \begin{cases} ET_j - ST_j, & \text{if } ET_i > ET_j \\ ET_i - ST_j, & \text{if } ET_i < ET_j \\ ET_j - ST_j, & \text{if } ET_i = ET_j \end{cases}$$

6.4.4 Task Switching and Multitasking Metrics

Metrics are required to gain a statistical understanding of a particular waste. The different metrics associated with task switches and multitasking at the different level of analysis are:

1. **Same case - different activity frequency:** Number of times a resource performs task switching or multitasking between different activities within the same case.
2. **Different case - same activity frequency:** Number of times a resource performs task switching or multitasking between different cases and performs the same activity.
3. **Different case - different activity frequency:** Number of times a resource performs task switching or multitasking between different activities of different cases.
4. **Switch/multitask within log frequency:** Number of times a resource performs task switching or multitasking between different activities in a log.
5. **Total switch frequency:** Total number of times the resource has performed task switching i.e. Summation of same case - different activity switch, different case - same activity switch, and different case - different activity.
6. **Total multitasking frequency:** Total number of times the resource has performed multitasking i.e. Summation of the same case - different activity multitasking, different case - same activity multitasking, and different case - different multitasking.
7. **Concurrency time:** The time in which two or more activities are executed concurrently by the same resource during a task switch.

The total switch or multitasking frequency can be ranked in descending order to gain an overall understanding of the switches or multitasking performed by a resource. An an-

alyst can also rank the individual metrics associated with the different levels of analysis to gain a deeper understanding for a specific type of switch or multitasking.

6.5 Waiting

This section presents the steps to identify the waiting waste from event logs. Waiting is defined as a delay in the actions to achieve process transformation [13]. Based on the tag analysis presented in section 5.5, waiting time manifests itself in the form of bottlenecks. A bottleneck indicates an activity or machine that blocks or affects the execution of other activities [24]. The start and the end timestamps are required to identify the waiting time in a process. Furthermore, resource information is needed to gain analyze bottlenecks at the activity-resource level. The subsequent sections outline the different steps to identify waiting time and bottleneck in an event log.

6.5.1 Waiting and Bottleneck Preliminaries

Consider an event log L, such that each case C_n contains a set of events E_s such that $E_s = e_1, e_2, e_3 \dots e_i$, where i and n is a unique event and case identifier respectively. To identify waiting time, each event e_i should have a few required attributes namely an activity A, a start timestamp ST, and end timestamp ET.

$$attr(A \wedge ST \wedge ET) \in e_i \tag{35}$$

The waiting time between two activities WT_{a_i,a_j} is the difference between the start timestamp of the second activity ST_j and the end timestamp of the first activity ET_i.

$$WT_{a_i,a_j} = ST_j - ET_i \tag{36}$$

To identify bottleneck, the resource R attribute is needed for further analysis.

$$attr(A \wedge R \wedge ST \wedge ET) \in e_i \tag{37}$$

6.5.2 Waiting Identification

In a business process, waiting time occurs when an activity waits for a resource to become available [1]. Waiting time can be detected at the activity and case level.

Waiting time at Activity Level identifies the waiting time between two activities of a case present in a log. Algorithm 4 outlines the steps to identify waiting time at the activity level.

Algorithm 4: Waiting Time at Activity Level

Input: $LogL$
Result: WT_{act}
Steps:

1. Find the consecutive activity pairs $A_i \rightarrow A_j$ present in a log such that each pair belongs to a case i.e. two activities will not form a pair if the belong to different case.
2. Find the occurrence n of each activity pair in the log.
3. For each activity pair, find the sum of the waiting time of all the occurrences (WT_{act}).

$$WT_{act} = \sum_{x=1}^{n} [WT_{a_i,a_j}]_x \tag{38}$$

Waiting Time at Case Level is the waiting time of each case in a log. Thus, case waiting time (WT_{case}) is the sum of the waiting time between two activities A_i and A_j present in the case. Algorithm 5 outlines the steps to calculate waiting time at the case level.

Algorithm 5: Waiting Time at Case Level

Input: $LogL$
Result: WT_{case}
Steps:

1. Find the consecutive activity pairs $A_i \rightarrow A_j$ present in a case. Let the total number of activity pairs present in a case be x.
2. Find the occurrence n of each activity pair in the case.
3. For each activity pair, find the sum of the waiting time of all the occurrences (WT_{act}).

$$WT_{act} = \sum_{x=1}^{n} [WT_{a_i,a_j}]_x \tag{39}$$

4. Calculate the case waiting time WT_{case}.

$$WT_{case} = \sum_{i=1}^{x} [WT_{act}]_i \tag{40}$$

6.5.3 Bottleneck Identification

A *Bottleneck* is an activity that obstructs the execution of other activities [96]. Roser et al. [79] considered bottleneck as the machine with the longest active period. Algorithm 6 presents the steps to identify the bottleneck activity and the bottleneck resource.

Algorithm 6: Bottleneck at Activity-Resource Level

Input: $LogL$

Result: Bottleneck Resource BR

Steps:

1. Calculate the processing time of each activity A_i present in a case.

$$PT_{act} = ET_i - ST_i \tag{41}$$

2. Let x be the number of times activity A_i occurs in a log. Calculate the total processing time of the activity in a log (TPT_{act}).

$$TPT_{act} = \sum_{i=1}^{x} [PT_{act}]_i \tag{42}$$

3. Find the activity with the highest processing time in a log. This is the bottleneck activity BA
4. For the activity with the highest processing time BA
   (a) Find the different resources that performed the bottleneck activity BA.
   (b) For each resource R
      i. Calculate n where n is the number of times R has performed the bottleneck activity BA.
      ii. For each time the resource has performed BA, calculate the processing time PT_r. PT_r is the processing time of the bottleneck activity performed by the resource each time.

$$PT_r = ET_{ba} - ST_{ba} \tag{43}$$

      iii. Calculate the total processing time (TPT_r) of BA performed by a resource. TPT_r the summation of the processing time (PT_r) of BA performed by R.

$$TPT_r = \sum_{i=1}^{n} [PT_r]_i \tag{44}$$

   (c) Find the resource that takes the highest processing time to execute the bottleneck activity BA. This is the bottleneck resource BR.

The activity with the highest processing time is the bottleneck activity. The identification of the bottleneck activity is not sufficient to detect the root cause of that bottleneck.

Thus, to analyze further, the bottleneck resource is identified. The resource that takes the highest processing time to execute the bottleneck activity is known as the bottleneck resource.

6.5.4 Waiting Time and Bottleneck Metrics

The following metrics can be used to understand the impact of waiting time and bottlenecks in a process.

1. **Activity Waiting Time:** The waiting time between the different activity pairs present in a log.
2. **Case Waiting Time:** The waiting time of the different activity pairs in each case.
3. **Total Processing Time:** The total processing time of a resource R performing a bottleneck activity.

The analyst can rank the metrics in descending order and consider the topmost or the top 25% as the problematic activities or resource.

6.6 Overprocessing

This section includes the steps to detect overprocessing waste from event logs. Overprocessing refers to unnecessary perfectionism or performing unnecessary tasks [1]. Knockout checks can cause overprocessing [1]. The information of whether an activity is a knockout check or the different variants present in an event log is required to detect this waste.

6.6.1 Overprocessing Preliminaries

Consider an event log L, with V case variants. A case variant is a unique path from the start to the end of a process. Each case variant includes a set of activities A_s.

6.6.2 Overprocessing Identification

Overprocessing refers to work that is performed unnecessarily [1]. This also includes unnecessary perfectionism [1]. Thus, this can indicate that a case variant with more number of activity instances could be due to overprocessing of the activities. In addition to this, knockout checks lead to unnecessary processing of tasks. A knockout check is defined as an activity that classifies a case as accepted or rejected [1]. If the case is rejected, the activities performed before the knockout check becomes overprocessing waste as they were unnecessarily performed [1].

However, it is difficult to identify a knockout activity or the valid case variant within a process solely based on data. Therefore, a hybrid approach that combines both human input and data from event logs can facilitate the detection of overprocessing waste from an event log. To capture the aspect of unnecessary perfectionism, the trace deviation is calculated to identify the variants that include more number of activity instances as compared to the valid variants. On the other hand, to capture the cases rejected due to the presence of a knockout check, input from the domain expert should be taken to identify the activity that is a knockout check. Algorithm 7 outlines the steps to identify the case variants with more number of activity instances.

Algorithm 7: Overprocessing in Log

Input: Valid variant V
Result: Trace Deviation and Cycle Time
Steps:

1. Find the different variants in a log.
2. For each pair of variants (V and V_i) such that each pair includes the valid variant V
   (a) Find the trace deviation. The trace deviation is the difference between the activity transitions in both variants.
   (b) Calculate the difference in the cycle time of both the variants.

6.6.3 Overprocessing Metrics

Based on the algorithm, to understand the impact of overprocessing waste in a business process, the following metrics can be used:

1. **Trace Deviation:** The difference in the number of deviations between two variants.
2. **Cycle Time Difference:** The difference between the cycle time of two variants.

The variant pair with the maximum trace deviation indicates that more activities are performed in that variant than the valid variant. The cycle time difference can further help to identify the impact in terms of duration. The analyst can choose to rank these metrics in descending order to identify the highest trace deviation or cycle time between the variant pairs.

6.7 Overproduction

This section presents the steps to detect overproduction waste from event logs. Overproduction occurs when an entire process instance is executed and does not add any

value on completion [1]. It is possible to detect overproduction waste if an event log has cancellation information.

6.7.1 Overproduction Preliminaries

Consider an event log L, such that each case C_n contains a set of events E_s such that $E_s = e_1, e_2, e_3 \dots e_i$, where i and n is a unique event and case identifier respectively. To identify overproduction, one needs the cancellation attribute CL to be present in the log. Furthermore, the start and end timestamps are required to calculate the impact in terms of duration.

$$attr(A \wedge CL \wedge ST \wedge ET) \in e_i \tag{45}$$

6.7.2 Overproduction Identification

Overproduction refers to producing too much or too soon than required [9, 43, 46]. This involves capturing the information on the number of units produced and the demand for it. An event log captures the execution of a process. Therefore, it is possible to consider the output of the process (products produced). However, the demand for the product is not commonly captured in an event log. It would be possible to enrich an event log with data from sales. However, the scope of this thesis is of what is captured in an event log. Therefore, overproduction, manifested as supply not matched by demand, is out of the scope of this thesis.

However, another manifestation of overproduction can be cancellations. If a case has begun but been cancelled, it can be considered valid for overproduction waste. Since the case was cancelled, the output does not deliver any value and is considered as overproduction [1]. An event log commonly does not hold data to determine with certainty if a case has been cancelled or not. A case can, at any point, be concluded without all activities being executed. However, the reason might be other than cancellation. If an event log holds data on which cases are cancelled, it is possible to identify overproduction expressed as cancellations. In such cases, the cases are sorted according to the attribute noting if the case is cancelled or not. By separating cases that are cancelled from other ones, one can detect this kind of waste.

Algorithm 8: Overproduction in Log

Input: Log L
Result: Number of cases C_n and Cycle Time CT
Steps:

1. Find the number of cases C_n that have been rejected i.e includes the cancellations attribute.
2. Find the total cycle time CT of all the cancelled cases.

6.7.3 Overproduction Metrics

The different metrics to measure the impact of overproduction waste are:

1. **Number of cases:** The number of cases cancelled.
2. **Cycle time:** The total cycle time of all the cancelled cases.
3. **Score:** The multiplication of cycle time and number of cases.

An analyst can choose to rank the metrics in descending order based on the number of cases, the cycle time or the score to identify the impact of overproduction waste in a process.

6.8 Defects

This section presents the patterns to identify defect waste from event logs. Defect waste arises from work performed to correct, repair, or compensate for a defect in a process [1]. As mentioned in section 5.8, defects show up in the form of rework and scrap. In an event log, rework can be detected from self-loop and eventually follows loop [28, 96]. However, it is not possible to identify scrap from event logs because scrap is the excessive discarded material that is not usable anymore. An event log captures the process of a business and does not hold data of the physical storage. Thus, the information of these discarded materials is not captured in an event log. The subsequent sections present the patterns associated with different types of rework in an event log.

6.8.1 Rework Preliminaries

Consider an event log L, such that each case C_n contains a set of events E_s such that $E_s = e_1, e_2, e_3 \dots e_i$, where i and n is a unique event and case identifier respectively. To identify rework, each event e_i should have a few required attributes namely an activity A, a start timestamp ST, and end timestamp ET.

$$attr(A \wedge ST \wedge ET) \in e_i \tag{46}$$

6.8.2 Rework Identification

Rework manifests itself in the form of different loop patterns in a process [28]. Thus, rework is classified into two main types: *self-loop* and *eventually-follows loop*. A *self-loop* represents the consecutive repetition of the same activity. Activity instances of the same activity type executed immediately after each other forms a self-loop [96]. In an event log, a self-loop occurs when the same activity A_i is executed immediately after the previous execution. For each transition between events e_i and e_j, a self-loop occurs

when:

$$\begin{aligned} attr(A_i,\, ST_i,\, ET_i) \rightarrow attr(A_j,\, ST_j,\, ET_j) \\ where\, ET_i < ST_j, ET_j > ET_i, A_i = A_j \end{aligned} \tag{47}$$

Swennen et al. [24] further classified self-loops into two: *Repeat self-loop* and *Redo self-loop*. *Repeat self-loop* occurs when the same activity is executed by the same resource immediately after each other within the same case [24]. Thus, for identification of this type, the resource attribute R is required. For each transition between events e_i and e_j, a repeat self-loop occurs when:

$$\begin{aligned} attr(A_i,\, ST_i,\, ET_i,\, R_i,\, C_i) \rightarrow attr(A_j,\, ST_j,\, ET_j,\, R_j,\, C_j) \\ where\, ET_i < ST_j,\, ET_j > ET_i,\, A_i = A_j,\, R_i = R_j,\, C_i = C_j \end{aligned} \tag{48}$$

Redo self-loop occurs when the same activity is executed by two different resources immediately after each other within the same case [24]. For each transition between events e_i and e_j, a redo self-loop occurs when:

$$\begin{aligned} attr(A_i,\, ST_i,\, ET_i,\, R_i,\, C_i) \rightarrow attr(A_j,\, ST_j,\, ET_j,\, R_j,\, C_j) \\ where\, ET_i < ST_j,\, ET_j > ET_i,\, A_i = A_j,\, R_i \neq R_j,\, C_i = C_j \end{aligned} \tag{49}$$

An *eventually-follows* loop occurs when same activity is executed again after the execution of different activities [24]. This occurs when the same activity A_i is eventually executed again after the execution of other activities. At a minimum, for an eventually-follows loop to occur, an activity is performed again after the execution of another activity. Thus, for a transition between three events e_i, e_j, and e_k, an eventually-follows loop occurs when:

$$\begin{aligned} attr(A_i,\, ST_i,\, ET_i) \rightarrow attr(A_j,\, ST_j,\, ET_j) \rightarrow attr(A_k,\, ST_k,\, ET_k) \\ where\, ET_i < ST_j\,,\, ET_j < ST_k\,,\, ET_i < ST_k\,,\, A_i = A_k\,,\, A_i \neq A_j \end{aligned} \tag{50}$$

An eventually-follows loop can be further divided into *Repeat repetition* and *Redo repetition*. A *Repeat repetition* loop occurs when an activity is eventually performed by the same resource within the same case [24]. At a minimum, for a transition between three events e_i, e_j, and e_k, a repeat repetition occurs when:

$$\begin{aligned} attr(A_i,\, ST_i,\, ET_i,\, R_i,\, C_i) \rightarrow attr(A_j,\, ST_j,\, ET_j,\, R_j,\, C_j) \rightarrow attr(A_k,\, ST_k,\, ET_k,\, R_k,\, C_k) \\ where\, ET_i < ST_j,\, ET_j < ST_k,\, ET_i < ST_k,\, A_i = A_k,\, A_i \neq A_j,\, R_i = R_k,\, C_i = C_j = C_k \end{aligned} \tag{51}$$

Redo repetition occurs when an activity is eventually performed by different resources within the same case [24]. For a transition between three events e_i, e_j, and e_k, a redo repetition occurs when:

$$attr(A_i,\, ST_i,\, ET_i,\, R_i,\, C_i) \rightarrow attr(A_j,\, ST_j,\, ET_j,\, R_j,\, C_j) \rightarrow attr(A_k,\, ST_k,\, ET_k,\, R_k,\, C_k)$$
$$where\, ET_i < ST_j,\, ET_j < ST_k,\, ET_i < ST_k,\, A_i = A_k,\, A_i \neq A_j,\, R_i \neq R_k,\, C_i = C_j = C_k \tag{52}$$

6.8.3 Rework Metrics

Different metrics can be used to gain an overall statistical view of the different types of rework in a process. The various metrics that can be used to study the impact of rework are:

1. **Frequency of self-loop:** The number of times each activity is directly followed by itself in a log.
2. **Frequency of Redo self-loop:** The number of times each activity is directly followed by itself and executed by different resources in a log.
3. **Frequency of Repeat self-loop:** The number of times each activity is directly followed by itself and executed by the same resource in a log.
4. **Frequency of eventually-follows loop:** The number of times each activity is eventually followed by itself in a log.
5. **Frequency of redo repetition loop:** The number of times each activity is eventually followed by itself and executed by different resources in a log.
6. **Frequency of repeat repetition loop:** The number of times each activity is eventually followed by itself and executed by the same resource in a log.

An analyst can choose to rank the individual metrics in descending order to identify the most common activities involved in different types of loops.

7 Discussion

In this section, the results and the limitations are discussed. The research objective of this thesis was to identify the different manifestations of wastes in a business process, the different methods and metrics used to identify these wastes in a business process, and how these manifestations can be discovered from event logs. The data obtained from the SLR on wastes and business improvement opportunities were analyzed to answer the research questions. To further complement the extracted literature, specific forms of waste in a business process were searched to extract additional articles specific to wastes.

The first research question was about the different manifestations of wastes in a business process. In addition to the seven types of wastes as discussed in numerous papers [5, 6, 13], a few of the papers presented an additional waste such as unused employee creativity, wishful thinking, avoid decision making, and knowledge waste [4, 11, 99]. Furthermore, Emiliani [98] considered the eighth waste to be the behavioral waste which is defined as the human behavior that does not add any value. All the different types of wastes that are dependent on human variables are considered to be a part of the behavioral waste and are tagged and categorized accordingly when analyzing the results.

The second research question was about how are the manifestations of wastes identified in a business process. The wastes in a business process can be identified manually using the VSM, or WID method [8, 9, 18] or by using a semi-automated approach that includes the analyst using a process mining tool to discover the process model [10, 28].

The final research question was about how can these manifestations of waste be discovered from an event log. The results of tag analysis show that a few of the wastes are already detected from event logs. However, partial or no information is present on detecting the remaining wastes from event logs. For these wastes, the rules and patterns were formulated based on the definitions and examples. Table 23 shows the different types of wastes and whether there was complete information (C.I), partial information (P.I), or no information (N.I) on detecting the wastes from event logs. Research on detecting different types of batch processing, rework, and a few of the handoffs from event logs already exists [20, 23, 24, 28]. For handoffs, the discovery of additional attribute combinations led to the identification of additional handoffs. Limited information was available on extracting ping-pong, bottlenecks, and waiting time from event logs [22, 55, 79]. However, there were many definitions and examples discussed in various papers as mentioned in section 5.2 and 5.5.

Table 23: Types and Information on Detection from Event Log

Types of Wastes	**C.I**	**P.I**	**N.I**
Human to Human Handoff	✓		
System to System Handoff	✓		
System to Human Handoff		✓	
Human to System Handoff		✓	
Human(system) to Human(system) Handoff	✓		
Int. Human to Ext. Human Handoff		✓	
Int. Human to Ext. System Handoff		✓	
Int. System to Ext. Human Handoff		✓	
Int. System to Ext. System Handoff		✓	
Int. Human(system) to Ext. Human(system) Handoff		✓	
Ping-Pong		✓	
WIP			✓
Batch Processing	✓		
Task Switching			✓
Multitasking			✓
Waiting		✓	
Bottleneck		✓	
Overprocessing			✓
Overproduction			✓
Directly-follows Rework	✓		
Eventually-follows Rework	✓		

No information was present on the detection of WIP, multitasking, task switching, over-processing, and overproduction from event logs. In this case, the rules and patterns for detecting them were formulated based on the definitions. The formulated patterns and rules serve to identify the different expressions of waste and rank them according to their impact. Based on the results, the analyst or the domain expert is armed with relevant information and evidence extracted from data to decide on whether the highest source of inefficiency is valid or not.

7.1 Limitations

This section presents the limitations of this research. The limitations are identified by following the guidelines of identifying, categorizing, and mitigating threats to validity in software engineering secondary studies [100], and the issues and best practices in content analysis [101].

The main limitations were the validity of the study selection, and data [100]. The threat

to the *Study Selection Validity* includes the adequacy of initial relevant publication identification, paper inaccessibility, missing non-English papers, and inclusion/exclusion of study. The threat of inadequate identification of relevant publication data was mitigated by performing an additional search on Google Scholar because it contains the search metadata of literature across different publishing formats and includes grey literature such as conference or workshop proceedings. The formulated search strings were developed to cover information on specific wastes to complement the broad view obtained by the two selected SLRs. The threat of inclusion/exclusion of relevant studies was mitigated by double-checking the papers that were excluded based on the title and abstract by reading the introductions. However, some relevant studies might have been missed due to the exclusion of non-English and non-accessible papers. Since the number of non-English and non-accessible papers was relatively small (4 and 40 respectively) as compared to 644 papers, this did not significantly impact the results.

The limitations associated with the content analysis includes the sampling with keyword searches and reliability issue in content analysis. The threat of *Sampling with Keywords Searches* includes not using relevant search strings and databases. This was mitigated by formulating seven search strings to cover specific manifestations of waste information and not miss any additional relevant data.

The threat to the *Reliability* of content analysis is dependent on the coder's consistency to tag and categorize information. This threat is mitigated by following a coding protocol that results in the consistent categorization of information. Furthermore, the tags and categories were reviewed again after the final formation of the categories to double-check the consistency.

8 Conclusion

This research presents a framework that outlines the data-driven rules and patterns to detect wastes from event logs. This research aims to identify the different manifestations of waste in a business process, how these manifestations can be identified in a business process, and the different ways in which the manifestations of waste can be detected from event logs. A combination of inductive and deductive content analysis was performed to synthesize, refine, and map scattered information on wastes, process mining, and event logs.

A complete reference list was requested from the authors who performed an SLR on waste and business improvement opportunities. Furthermore, an additional search was performed to fetch information on specific wastes to ensure that relevant articles are not missed. After executing the selection procedure, a total of 187 papers were shortlisted for tagging and categorization. This process resulted in 1096 tags categorized into 23 subcategories and 9 main categories.

The 9 main categories identified from tagging the information present in 187 papers were: waste, transportation, inventory, motion, waiting, overprocessing, overproduction, defects, and behavioral. Each category was further divided into an overview subcategory covering the definition, description, examples, possible causes, and relation of the waste category with other wastes. The remaining subcategories mainly cover the different ways in which a waste expresses itself. Based on the different expressions of waste discovered from the tag analysis, a framework was formulated to outline the data-driven rules and patterns to identify the seven wastes and their expression from event logs. Since the behavioral category captures human behavior and cannot be detected from event logs, it wasn't included in the framework.

The different threats to the validity of the research were the validity of the study selection and data. The inadequate identification of relevant publications, inaccessibility of papers, missing non-English papers, inclusion/exclusion of studies, and reliability issues in the content analysis were the limitations. However, different mitigation measures were incorporated to minimize or eliminate these threats.

The different patterns and rules used to identify the expressions of wastes from event logs are derived from the different methodologies or case studies presented in various papers. Since the framework is conceptually derived, an important avenue of future research would be to implement the rules outlined in the framework to discover the different wastes from event logs.

References

[1] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers, *et al.*, *Fundamentals of business process management*. Springer, 2013, vol. 1.

[2] M. Hammer and J. Champy, *Reengineering the corporations*, 1993.

[3] L. Reinkemeyer, *Process Mining in Action*. Springer, 2020.

[4] T. Sedano, P. Ralph, and C. Péraire, "Software development waste", in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, IEEE, 2017, pp. 130–140.

[5] M. Ikonen, P. Kettunen, N. Oza, and P. Abrahamsson, "Exploring the sources of waste in kanban software development projects", in *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, IEEE, 2010, pp. 376–381.

[6] T. Ohno, *Toyota production system: beyond large-scale production*. crc Press, 1988.

[7] N. Modig and P. Åhlström, *This is lean: Resolving the efficiency paradox*. Rheologica, 2012.

[8] M. Rother and J. Shook, *Learning to see: value stream mapping to add value and eliminate muda*. Lean Enterprise Institute, 2003.

[9] J. Dinis-Carvalho, F. Moreira, S. Bragança, E. Costa, A. Alves, and R. Sousa, "Waste identification diagrams", *Production Planning & Control*, vol. 26, no. 3, pp. 235–247, 2015.

[10] Apromore, "Apromore documentation", 2020. [Online]. Available: `https://documentation.apromore.org/index.html` (visited on 04/22/2010).

[11] H. Alahyari, T. Gorschek, and R. B. Svensson, "An exploratory study of waste in software development organizations using agile or lean approaches: A multiple case study at 14 organizations", *Information and Software Technology*, vol. 105, pp. 78–94, 2019.

[12] B. J. Hicks, "Lean information management: Understanding and eliminating waste", *International journal of information management*, vol. 27, no. 4, pp. 233–249, 2007.

[13] M. Thürer, I. Tomašević, and M. Stevenson, "On the meaning of 'waste': Review and definition", *Production Planning & Control*, vol. 28, no. 3, pp. 244–255, 2017.

[14] J. Sutherland and B. Bennett, "The seven deadly wastes of logistics: Applying toyota production system principles to create logistics value", *White Paper No*, vol. 701, 2007.

[15] S. Dennis, B. King, M. Hind, and S. Robinson, "Applications of business process simulation and lean techniques in british telecommunications plc", in *2000 Winter Simulation Conference Proceedings (Cat. No. 00CH37165)*, IEEE, vol. 2, 2000, pp. 2015–2021.

[16] S. M. Dahlgaard-Park, M. F. Suárez-Barraza, J. Ramis-Pujol, and M. Estrada-Robles, "Applying gemba-kaizen in a multinational food company: A process innovation framework", *International Journal of Quality and Service Sciences*, 2012.

[17] W. K.-C. Yung, "A stepped composite methodology to redesign manufacturing processes through re-engineering and benchmarking", *International Journal of Operations & Production Management*, 1997.

[18] J. Dinis-Carvalho, R. C. Ratnayake, and L. Ferrete, "Implementation of lean principles for performance improvement: Use of vsm+ wid for waste identification", in *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, IEEE, 2017, pp. 644–648.

[19] D. Buchanan and B. Wilson, "Re-engineering operating theatres: The perspective assessed", *Journal of management in medicine*, 1996.

[20] B. T. Pentland, J. Recker, and G. Wyner, "Rediscovering handoffs", *Academy of Management Discoveries*, vol. 3, no. 3, pp. 284–301, 2017.

[21] I. Verenich, M. Dumas, M. La Rosa, F. M. Maggi, and C. Di Francescomarino, "Minimizing overprocessing waste in business processes via predictive activity ordering", in *International Conference on Advanced Information Systems Engineering*, Springer, 2016, pp. 186–202.

[22] P. Delias, "A positive deviance approach to eliminate wastes in business processes: The case of a public organization", *Industrial Management & Data Systems*, 2017.

[23] N. Martin, L. Pufahl, and F. Mannhardt, "Detection of batch activities from event logs", *Information Systems*, vol. 95, p. 101 642, 2021.

[24] M. Swennen, N. Martin, G. Janssenswillen, M. Jans, B. Depaire, A. Caris, and K. Vanhoof, "Capturing resource behaviour from event logs.", in *SIMPDA*, 2016, pp. 130–134.

[25] M. K. Hassan, "Applying lean six sigma for waste reduction in a manufacturing environment", *American Journal of Industrial Engineering*, vol. 1, no. 2, pp. 28–35, 2013.

[26] R. Mans, H. Schonenberg, G. Leonardi, S. Panzarasa, A. Cavallini, S. Quaglini, and W. Van Der Aalst, "Process mining techniques: An application to stroke care", in *MIE*, vol. 136, 2008, pp. 573–578.

[27] M. Leemans and W. M. van der Aalst, "Discovery of frequent episodes in event logs", in *International symposium on data-driven process discovery and analysis*, Springer, 2014, pp. 1–31.

[28] Disco, "Disco documentation", 2020. [Online]. Available: `https://fluxicon.com/blog/2017/03/how-to-identify-rework-in-your-process/` (visited on 04/30/2010).

[29] I. A. Rawabdeh, "A model for the assessment of waste in job shop environments", *International Journal of Operations & Production Management*, 2005.

[30] A. Kumar and B. Thummadi, "A sequence analytics approach for detecting handoff patterns in workflows: An exploratory case study on the volvo it incident management process", in *Designing the Digital Transformation: DESRIST 2017 Research in Progress Proceedings of the 12th International Conference on Design Science Research in Information Systems and Technology. Karlsruhe, Germany. 30 May-1 Jun.*, Karlsruher Institut für Technologie (KIT), 2017, pp. 11–19.

[31] J. F. DeFranco and P. A. Laplante, "A content analysis process for qualitative software engineering research", *Innovations in Systems and Software Engineering*, vol. 13, no. 2, pp. 129–141, 2017.

[32] S. Seuring and S. Gold, "Conducting content-analysis based literature reviews in supply chain management", *Supply Chain Management: An International Journal*, 2012.

[33] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering–a systematic literature review", *Information and software technology*, vol. 51, no. 1, pp. 7–15, 2009.

[34] K. Lashkevich, "Business process improvement opportunities: A framework to support business process redesign", 2020.

[35] M. Poppendieck and M. A. Cusumano, "Lean software development: A tutorial", *IEEE software*, vol. 29, no. 5, pp. 26–32, 2012.

[36] B. W. Oppenheim, "Lean product development flow", *Systems engineering*, vol. 7, no. 4, no–no, 2004.

[37] R. Singh, S. Kumar, A. Choudhury, and M. Tiwari, "Lean tool selection in a die casting unit: A fuzzy-based decision support heuristic", *International journal of production research*, vol. 44, no. 7, pp. 1399–1429, 2006.

[38] S. Gupta, M. Sharma, *et al.*, "Lean services: A systematic review", *International Journal of productivity and performance management*, 2016.

[39] M. U. Maldonado, M. E. Leusin, T. C. de Albuquerque Bernardes, and C. R. Vaz, "Similarities and differences between business process management and lean management", *Business Process Management Journal*, 2020.

[40] E. Kupiainen, M. V. Mäntylä, and J. Itkonen, "Using metrics in agile and lean software development–a systematic literature review of industrial studies", *Information and Software Technology*, vol. 62, pp. 143–163, 2015.

[41] R. J. Schonberger, "The transfer of japanese manufacturing management approaches to us industry", *Academy of Management Review*, vol. 7, no. 3, pp. 479–487, 1982.

[42] T. Falk, P. Griesberger, and S. Leist, "Patterns as an artifact for business process improvement-insights from a case study", in *International Conference on Design Science Research in Information Systems*, Springer, 2013, pp. 88–104.

[43] P. Lewis and G. Cooke, "Developing a lean measurement system to enhance process improvement", *International Journal of Metrology and Quality Engineering*, vol. 4, no. 3, pp. 145–151, 2013.

[44] E. Sandberg and L. Bildsten, "Coordination and waste in industrialised housing", *Construction Innovation*, 2011.

[45] R. Gerth, A. Boqvist, M. Bjelkemyr, and B. Lindberg, "Design for construction: Utilizing production experiences in development", *Construction Management and Economics*, vol. 31, no. 2, pp. 135–150, 2013.

[46] M. Adams, P. Componation, H. Czarnecki, and B. J. Schroer, "Simulation as a tool for continuous process improvement", in *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future-Volume 1*, 1999, pp. 766–773.

[47] A. Bonaccorsi, G. Carmignani, F. Zammori, *et al.*, "Service value stream management (svsm): Developing lean thinking in the service industry", *Journal of Service Science and management*, vol. 4, no. 04, p. 428, 2011.

[48] D. F. Garrett and J. Lee, "Lean construction submittal process—a case study", *Quality Engineering*, vol. 23, no. 1, pp. 84–93, 2010.

[49] R. V. B. de Souza and L. C. R. Carpinetti, "A fmea-based approach to prioritize waste reduction in lean implementation", *International Journal of Quality & Reliability Management*, 2014.

[50] S. Liu, M. Leat, J. Moizer, P. Megicks, and D. Kasturiratne, "A decision-focused knowledge management framework to support collaborative decision making for lean supply chain management", *International Journal of Production Research*, vol. 51, no. 7, pp. 2123–2137, 2013.

[51] R. Singh, A. Choudhury, M. Tiwari, and R. Maull, "An integrated fuzzy-based decision support system for the selection of lean tools: A case study from the steel industry", *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 220, no. 10, pp. 1735–1749, 2006.

[52] S. Ramakrishnan, S. Kumaran, H. Chang, N. Kulkarni, and K. Srihari, "Defining and categorizing handoff points for the service domain", in *Proceedings of the 29th Annual Conference of American Society for Engineering Management*, 2008, pp. 12–15.

[53] W. M. Van Der Aalst, H. A. Reijers, A. J. Weijters, B. F. van Dongen, A. A. De Medeiros, M. Song, and H. Verbeek, "Business process mining: An industrial application", *Information Systems*, vol. 32, no. 5, pp. 713–732, 2007.

[54] M. Rossi, E. T. Kerga, M. Taisch, and S. Terzi, "Proposal of a method to systematically identify wastes in new product development process", in *2011 17th International Conference on Concurrent Enterprising*, IEEE, 2011, pp. 1–9.

[55] J. Chongwatpol and R. Sharda, "Achieving lean objectives through rfid: A simulation-based assessment", *Decision sciences*, vol. 44, no. 2, pp. 239–266, 2013.

[56] P. Hines and N. Rich, "The seven value stream mapping tools", *International journal of operations & production management*, 1997.

[57] B. Poksinska, "The current state of lean implementation in health care: Literature review", *Quality management in healthcare*, vol. 19, no. 4, pp. 319–329, 2010.

[58] K. Petersen and C. Wohlin, "Software process improvement through the lean measurement (spi-leam) method", *Journal of systems and software*, vol. 83, no. 7, pp. 1275–1287, 2010.

[59] O. Al-Baik and J. Miller, "Waste identification and elimination in information technology organizations", *Empirical Software Engineering*, vol. 19, no. 6, pp. 2019–2061, 2014.

[60] I. Sitova and J. Pecerska, "Process data analysis using visual analytics and process mining techniques", in *2020 61st International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS)*, IEEE, 2020, pp. 1–6.

[61] E. Tribelsky and R. Sacks, "Measuring information flow in the detailed design of construction projects", *Research in engineering design*, vol. 21, no. 3, pp. 189–206, 2010.

[62] M. A. Quddus and A. N. Ahsan, "A shop-floor kaizen breakthrough approach to improve working environment and productivity of a sewing floor in rmg industry", *Journal of Textile and Apparel, Technology and Management*, vol. 8, no. 4, 2014.

[63] C.-C. Huang* and S.-H. Liu, "A novel approach to lean control for taiwan-funded enterprises in mainland china", *International Journal of Production Research*, vol. 43, no. 12, pp. 2553–2575, 2005.

[64] C. Rincon-Gonzalez and F. N. Diaz-Piraquive, "Scientometric analysis of knowledge in the context of project management", in *International Conference on Knowledge Management in Organizations*, Springer, 2019, pp. 14–24.

[65] N. Martin, M. Swennen, B. Depaire, M. Jans, A. Caris, and K. Vanhoof, "Batch processing: Definition and event log identification.", in *SIMPDA*, 2015, pp. 137–140.

[66] J. Nakatumba and W. M. van der Aalst, "Analyzing resource behavior using process mining", in *International Conference on Business Process Management*, Springer, 2009, pp. 69–80.

[67] L. Pufahl, N. Herzberg, A. Meyer, and M. Weske, "Flexible batch configuration in business processes based on events", in *International Conference on Service-Oriented Computing*, Springer, 2014, pp. 63–78.

[68] L. Pufahl and M. Weske, "Enabling batch processing in bpmn processes.", in *BPM (Demos)*, Citeseer, 2016, pp. 28–33.

[69] N. Martin, M. Swennen, B. Depaire, M. Jans, A. Caris, and K. Vanhoof, "Retrieving batch organisation of work insights from event logs", *Decision Support Systems*, vol. 100, pp. 119–128, 2017.

[70] H.-P. Tserng, S. Y.-L. Yin, and T.-L. Ngo, "A lean prebid planning model for construction contractors: A case study in vietnam", *Journal of Marine Science and Technology*, vol. 21, no. 4, pp. 430–441, 2013.

[71] T. Jylhä and S. Junnila, "The state of value creation in the real-estate sector–lessons from lean thinking", *Property Management*, 2014.

[72] L. M. Skaugset, S. Farrell, M. Carney, M. Wolff, S. A. Santen, M. Perry, and S. J. Cico, "Can you multitask? evidence and limitations of task switching and multitasking in emergency medicine", *Annals of emergency medicine*, vol. 68, no. 2, pp. 189–195, 2016.

[73] M. Korkala and F. Maurer, "Waste identification as the means for improving communication in globally distributed agile software development", *Journal of Systems and Software*, vol. 95, pp. 122–140, 2014.

[74] J. A. Elfving, I. D. Tommelein, and G. Ballard, "Consequences of competitive bidding in project-based production", *Journal of Purchasing and Supply Management*, vol. 11, no. 4, pp. 173–181, 2005.

[75] R. Arbulu, I. Tommelein, K. Walsh, and J. Hershauer, "Value stream analysis of a re-engineered construction supply chain", *Building Research & Information*, vol. 31, no. 2, pp. 161–171, 2003.

[76] N. Gautam, R. B. Chinnam, and N. Singh, "Design reuse framework: A perspective for lean development", *International Journal of Product Development*, vol. 4, no. 5, pp. 485–507, 2007.

[77] R. C. Dart, "Can lean thinking transform american health care?", *Annals of emergency medicine*, vol. 57, no. 3, pp. 279–281, 2011.

[78] L. Li, Q. Chang, and J. Ni, "Data driven bottleneck detection of manufacturing systems", *International Journal of production research*, vol. 47, no. 18, pp. 5019–5036, 2009.

[79] C. Roser, M. Nakano, and M. Tanaka, "Shifting bottleneck detection", in *Proceedings of the Winter Simulation Conference*, IEEE, vol. 2, 2002, pp. 1079–1086.

[80] M. Subramaniyan, A. Skoogh, H. Salomonsson, P. Bangalore, M. Gopalakrishnan, and A. Sheikh Muhammad, "Data-driven algorithm for throughput bottle-

neck analysis of production systems", *Production & Manufacturing Research*, vol. 6, no. 1, pp. 225–246, 2018.

[81] C. Roser, M. Nakano, and M. Tanaka, "A practical bottleneck detection method", in *Proceeding of the 2001 Winter Simulation Conference (Cat. No. 01CH37304)*, IEEE, vol. 2, 2001, pp. 949–953.

[82] R. F. Aziz and S. M. Hafez, "Applying lean thinking in construction and performance improvement", *Alexandria Engineering Journal*, vol. 52, no. 4, pp. 679–695, 2013.

[83] T. R. Rohleder and E. A. Silver, "A tutorial on business process improvement", *Journal of Operations Management*, vol. 15, no. 2, pp. 139–154, 1997.

[84] W. M. van der Aalst, "Re-engineering knock-out processes", *Decision Support Systems*, vol. 30, no. 4, pp. 451–468, 2001.

[85] A. M. Deif, "Dynamic analysis of a lean cell under uncertainty", *International Journal of Production Research*, vol. 50, no. 4, pp. 1127–1139, 2012.

[86] D. F. Mathaisel, "A lean architecture for transforming the aerospace maintenance, repair and overhaul (mro) enterprise", *International Journal of Productivity and Performance Management*, 2005.

[87] L. S. Pheng, F. M. Arain, and J. W. Y. Fang, "Applying just-in-time principles in the delivery and management of airport terminal buildings", *Built Environment Project and Asset Management*, 2011.

[88] J. MacBryde, Z. Radnor, S. Taj, and L. Berro, "Application of constrained management and lean manufacturing in developing best practices for productivity improvement in an auto-assembly plant", *International Journal of Productivity and Performance Management*, 2006.

[89] D. T. Jones, P. Hines, and N. Rich, "Lean logistics", *International Journal of physical distribution & logistics management*, 1997.

[90] S. Vinodh, S. Gautham, and A. Ramiya R, "Implementing lean sigma framework in an indian automotive valves manufacturing organisation: A case study", *Production Planning & Control*, vol. 22, no. 7, pp. 708–722, 2011.

[91] G. K. Hansen and N. O. Olsson, "Layered project–layered process: Lean thinking and flexible solutions", *Architectural Engineering and Design Management*, vol. 7, no. 2, pp. 70–84, 2011.

[92] P. E. Love, "Auditing the indirect consequences of rework in construction: A case based approach", *Managerial auditing journal*, 2002.

[93] A. G. Cass, L. J. Osterweil, and A. Wise, "A pattern for modeling rework in software development processes", in *International Conference on Software Process*, Springer, 2009, pp. 305–316.

[94] R. E. Fairley and M. J. Willshire, "Iterative rework: The good, the bad, and the ugly", *Computer*, vol. 38, no. 9, pp. 34–41, 2005.

[95] A. G. Cass, S. M. Sutton, and L. J. Osterweil, "Formalizing rework in software processes", in *European Workshop on Software Process Technology*, Springer, 2003, pp. 16–31.

[96] M. Swennen, G. Janssenswillen, M. Jans, B. Depaire, and K. Vanhoof, "Capturing process behavior with log-based process metrics.", in *SIMPDA*, 2015, pp. 141–144.

[97] P. Biswas and B. R. Sarker, "Optimal batch quantity models for a lean production system with in-cycle rework and scrap", *International Journal of Production Research*, vol. 46, no. 23, pp. 6585–6610, 2008.

[98] M. Emiliani, "Redefining the focus of investment analysts", *The TQM Magazine*, 2001.

[99] Y. Sugimori, K. Kusunoki, F. Cho, and S. UCHIKAWA, "Toyota production system and kanban system materialization of just-in-time and respect-for-human system", *The international journal of production research*, vol. 15, no. 6, pp. 553–564, 1977.

[100] A. Ampatzoglou, S. Bibi, P. Avgeriou, M. Verbeek, and A. Chatzigeorgiou, "Identifying, categorizing and mitigating threats to validity in software engineering secondary studies", *Information and Software Technology*, vol. 106, pp. 201–230, 2019.

[101] S. Lacy, B. R. Watson, D. Riffe, and J. Lovejoy, "Issues and best practices in content analysis", *Journalism & Mass Communication Quarterly*, vol. 92, no. 4, pp. 791–811, 2015.

Appendix

I. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Shefali Ajit Sharma**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Waste Identification from Event Logs,

   supervised by Fredrik Milani.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Shefali Ajit Sharma
14/05/2021