CHALMERS
UNIVERSITY OF TECHNOLOGY

Enabling Opportunistic Maintenance in the Sawmill Industry

LSTM-Based Prediction of Maintenance Opportunity Windows Using Discrete Event Simulation

Master's thesis in Production Engineering

DIEGO PÉREZ OLIVER

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE

DIVISION OF PRODUCTION SYSTEMS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Enabling Opportunistic Maintenance in the Sawmill Industry

LSTM-Based Prediction of Maintenance Opportunity Windows Using Discrete Event Simulation

DIEGO PÉREZ OLIVER

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Industrial and Materials Science
Division of Production Systems
Chalmers University of Technology
Gothenburg, Sweden

Enabling Opportunistic Maintenance in the Sawmill Industry

LSTM-Based Prediction of Maintenance Opportunity Windows Using Discrete Event Simulation

DIEGO PÉREZ OLIVER

Supervisor: Siyuan Chen, Industrial and Materials Science

Examiner: Anders Skoogh, Industrial and Materials Science

Master's thesis 2026

Department of INDUSTRIAL AND MATERIALS SCIENCE

Division of PRODUCTION SYSTEMS

Chalmers University of Technology

SE-412 96 Gothenburg

Telephone +34 691 165 880

Gothenburg, Sweden 2026

Enabling Opportunistic Maintenance in the Sawmill Industry

LSTM-Based Prediction of Maintenance Opportunity Windows Using Discrete Event Simulation

DIEGO PÉREZ OLIVER

Department of Industrial and Materials Science

Chalmers University of Technology

Abstract

The sawmill industry has traditionally fell behind other manufacturing sectors in the adoption of modern maintenance and asset management practices, particularly regarding predictive and opportunistic maintenance strategies. This challenge is compounded by the limited amount of scientific literature and industrial case studies focusing specifically on sawmill environments. At the same time, this gap presents an opportunity to improve productivity and create competitive advantage through the implementation of advanced maintenance approaches.

This study is conducted in collaboration with Södra Skogsägarna and investigates the feasibility of introducing opportunistic maintenance within the Swedish sawmill industry through the prediction of Maintenance Opportunity Windows (MOWs). The project builds upon recent research on MOW prediction and aims to adapt these concepts to the sawmill production context using Long Short-Term Memory (LSTM) networks.

Since sufficient production data and data collection infrastructure are currently unavailable, a Discrete-Event Simulation model of a generic sawmill will be developed to generate synthetic production data. The simulation model will support an iterative process for identifying relevant input variables and defining future data collection requirements for real-world implementation.

In addition to enabling the development of the prediction model, the simulation environment will also be used to evaluate the impact of different production policies and external factors on system performance. The study aims to contribute both to the academic understanding of opportunistic maintenance in sawmill systems and to the practical implementation of data-driven maintenance strategies in industry.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Siyuan Chen, for his continuous guidance, valuable feedback, and support throughout this thesis. His advice and constructive comments have been essential to the development of this work.

I would also like to thank my examiner, Anders Skoogh, for his valuable feedback and for providing me with the opportunity to participate in this research project. His insights and support have been greatly appreciated throughout the course of the study.

I would like to express my sincere gratitude to Professor Adolfo Crespo Márquez for his continued support and guidance throughout my academic journey. His enthusiasm for the field of maintenance engineering and research played a significant role in motivating me to pursue both this area of study and a scientific career. His mentorship and encouragement over the years have been invaluable.

Special thanks are extended to Chalmers e-Commons at Chalmers for providing the computational resources required to perform the machine learning experiments presented in this work.

I am grateful to Södra for enabling this project and for providing access to the industrial context that made this research possible. In particular, I would like to thank Tinh Sjökvist for serving as company supervisor and for sharing valuable insights regarding sawmill operations, maintenance practices, and working conditions. I would also like to thank all personnel who contributed their time and knowledge throughout the project.

I would further like to thank the members of the Ready 2 project for their collaboration and support. In particular, I am grateful to Rashid Ali for his valuable insights and discussions regarding machine learning and LSTM models.

Finally, I would like to thank Mukund for taking the time to clarify details of the previously developed Maintenance Opportunity Window prediction model, which provided important guidance during this research.

Göteborg, June 2026
Diego Pérez Oliver

List of Acronyms

AM	Autonomous Maintenance
CBM	Condition-Based Maintenance
CM	Corrective Maintenance
CMMS	Computerised Maintenance Management System
DES	Discrete-Event Simulation
ERP	Enterprise Resource Planning
HMI	Human Machine Interface
LIFO	Last-In, First-Out
LSTM	Long Short-Term Memory
ML	Machine Learning
MOW	Maintenance Opportunity Window
MTBF	Mean Time Between Failures
MTTR	Mean Time to Repair
OM	Opportunistic Maintenance
PdM	Predictive Maintenance
PM	Preventive Maintenance
RNG	Random Number Generator
TPM	Total Productive Maintenance

Table of Contents

List of figures

List of tables

1
Introduction

This thesis investigates possible improvements in maintenance strategies within the sawmill industry, with a particular focus on the identification and utilisation of Maintenance Opportunity Windows (MOWs). By leveraging data-driven approaches and simulation-based analysis, the study aims to enhance production efficiency and reduce downtime. The research is conducted in collaboration with Södra, providing a real-world industrial context for the proposed methods.

1.1 Industry Background

Södra is a Swedish forest industry group, with products ranging from biochemicals to wood-based products. Sawmills constitute a key part of their business operations and play a crucial role in the supply chain for numerous products and companies (Södra skogsägarna, n.d.). The company aims to upgrade its maintenance strategy to remain competitive and current, moving away from traditional approaches.

Inadequate maintenance in this sector can be particularly critical, as much of the equipment in a sawmill is hazardous and poses risks to operators. For example, saw blades are high-speed rotating tools that, although usually protected by shields, can cause kickbacks if not properly maintained. Similarly, poorly maintained electrical installations may result in catastrophic events, such as fires in areas containing flammable materials.

Moreover, a lack of modern production practices is noticeable within the industry, particularly the prevalence of large buffer levels, which carry an inherent risk of material degradation over time.

Despite these risks, formal studies on maintenance practices in the wood industry – and sawmills specifically – remain limited. The sector traditionally relies on reactive maintenance supplemented with preventive measures, often requiring production shutdowns. While this presents a significant opportunity for improvement, it also entails a steep learning curve for implementing advanced techniques, particularly in contexts where proper data collection is not yet established. Additionally, the organisation may need to undergo a cultural shift, potentially facing resistance from personnel, as highlighted by Burnes (1996).

The READY2 project, where Chalmers participates, aims to build a sustainable, robust and resource-efficient wood manufacturing industry by

transforming maintenance practices and introducing data-drive approaches. This creates a suitable environment to develop this thesis.

1.2 Research Background

1.2.1 Evolution of Maintenance Strategies

Maintenance practices have evolved considerably throughout the history of manufacturing. Early maintenance strategies were predominantly reactive, focusing on restoring equipment functionality after failures occurred. However, advances in sensing technologies, data collection systems, computing power, and analytical methods have enabled increasingly proactive approaches to maintenance management, as shown by Garg & Deshmukh (2006).

As a consequence, research and industrial practice have progressively shifted towards maintenance strategies that seek to anticipate failures rather than react to them. These approaches include preventive, condition-based, predictive, and prescriptive maintenance strategies, each relying on increasing levels of information and decision support. Numerous studies have reported that proactive maintenance approaches can improve equipment availability, production stability, and overall organisational performance when compared with purely reactive strategies (Pintelon et al., 2006).

The increasing adoption of data-driven technologies has further accelerated this transition, creating opportunities to integrate maintenance decisions with production planning and operational management. Within this broader evolution, opportunistic maintenance has emerged as a promising approach for reducing the production impact of maintenance activities.

1.2.2 Opportunistic maintenance and Maintenance Opportunity Windows

Opportunistic maintenance is a maintenance strategy that can reduce downtime and increase productivity (Bokrantz et al., 2025; Colledani et al., 2018). Based on case data provided by Södra, which suggests potential losses in sold value due to maintenance inefficiencies, implementing opportunistic maintenance could yield a substantial revenue increase for the production site.

Opportunistic maintenance is gaining increasing attention in the academic field. One recent research direction presented by Bokrantz et al. (2025) concerns the identification and prediction of Maintenance Opportunity Windows (MOWs), periods during which maintenance can be performed with limited impact on production throughput.

Predicting MOWs opens a broader range of research opportunities aimed at improving productivity through maintenance optimisation. Vu et al. (2020) synthetise a categorisation based on Nicolai & Dekker (2008), identifying three types of dependencies that can create maintenance opportunities in multi-

component systems. Both Iung et al. (2016) and Vu et al. 2020) apply genetic algorithms to group maintenance activities according to cost-optimisation criteria.

1.2.3 Research gap

Most of the existing literature focuses on how maintenance opportunities can be exploited once they have been identified, rather than on how such opportunities can be predicted. Research addressing MOW prediction remains limited. For example, the approach proposed by Bokrantz et al. (2025) focuses on the prediction of fixed-length opportunity windows. Several limitations remain and create opportunities for further research, particularly regarding the prediction of variable-length windows and the identification of consecutive opportunities that could be combined to accommodate longer maintenance activities.

Furthermore, the reviewed studies primarily focus on maintenance scheduling and optimisation while assuming the availability of the necessary maintenance resources. In practice, the availability of maintenance personnel, tools, and spare parts may significantly influence whether a maintenance opportunity can actually be exploited. Incorporating such constraints could improve the realism and practical applicability of opportunistic maintenance approaches, reducing the gap between maintenance planning and operational implementation.

The limited body of research addressing MOW prediction, together with the absence of studies explicitly considering maintenance-resource constraints in a sawmill context, motivates the present study.

1.3 Aim & Purpose

Maintenance-related production losses continue to represent a significant challenge for manufacturing companies (Thomas & Weiss, 2021). While proactive maintenance strategies can improve equipment availability and production performance, their implementation often requires maintenance activities to be carefully coordinated with production requirements. Opportunistic Maintenance offers a promising approach by exploiting naturally occurring interruptions in production to perform maintenance without introducing additional downtime. However, the practical application of this strategy depends on the ability to identify or anticipate suitable Maintenance Opportunity Windows (MOWs).

Although research concerning Opportunistic Maintenance has grown in recent years, relatively limited attention has been devoted to the prediction of MOWs before they occur. Furthermore, existing work has been developed and validated in industrial contexts that differ substantially from the sawmill industry.

The primary aim of this thesis is to develop a model capable of predicting Maintenance Opportunity Windows (MOWs). A secondary objective is to

evaluate lean production scenarios and provide actionable insights for the company. The work is also expected to contribute to a scientific publication.

The company recognises production habits that can be regarded as outdated. There is evidence that more recent procedures lead to enhanced performance, and there is willingness to invest in methods based in recent trends that can improve the company's competitiveness. In this thesis, the main focus is placed upon maintenance improvement, with a lesser focus in the evaluation of scenarios more inclined towards lean manufacturing.

1.3.1 Primary aim – MOWs prediction model

The primary goal of this thesis is to enable opportunistic maintenance through the prediction of upcoming MOWs. The work builds on the approach proposed by Bokrantz et al. (2025) that employs a Long Short-Term Memory (LSTM) model for prediction. Furthermore, the study seeks to evaluate the applicability of the proposed approach in a different industrial context. While the previous paper was developed and its methodology validated within the automotive industry, the present work investigates its feasibility in a sawmill environment, characterised by different production processes, material flows, and operational conditions

However, several improvements to the existing model are proposed. Instead of predicting fixed-length windows, the model could be extended to identify windows of varying duration that meet a minimum threshold. Additionally, the theoretical framework could be expanded to incorporate further system constraints and contextual variables, enriching the MOW concept.

This objective represents the main scientific contribution of the thesis and is directly linked to the research questions outlined below.

1.3.2 Secondary aim – Scenario Evaluation

Preliminary findings indicate a lack of lean practices within the sawmill industry, particularly regarding buffer sizes. Given the scope of the thesis, it was considered valuable to provide insights into scenarios involving buffer size reduction and their impact on production performance.

Materials stored in these buffers, such as unprocessed logs, are subject to degradation over time, leading to reduced quality or material loss. Additionally, the sector is currently facing material supply constraints due to factors such as wildfires and policy changes (Eliasson, 2026; Swedish Forest Industries Federation, 2025). These scenarios will therefore also evaluate the impact of supply shortages.

Finally, extreme production scenarios requested by the company will be analysed. These include cases where one product type is processed fastest in one machine but slowest in another, as well as the inverse situation. These cases aim to stress-test the system and evaluate its behaviour under atypical conditions.

1.4 Research Questions

Maintenance Opportunity Windows (MOWs) are defined as time periods that occur within production and can be utilised for maintenance activities without affecting system throughput. However, it remains unclear whether all such windows are practically feasible for maintenance, and no clear criteria for feasibility have been established. Additionally, there is a lack of publications focusing on maintenance within the sawmill industry, and according to industry practitioners, the sector falls behind in current maintenance trends. Finally, data collection systems are currently limited within the sawmill, hindering the practical implementation of data-based models. Although there is a willingness to invest in such capabilities, the minimum data requirements for effective MOW prediction remain unclear and should be addressed before any investments are made.

1. What constraints have to be met for a MOW to be considered valid in a sawmill context?
2. How can MOWs be predicted in a sawmill industry context?
3. How can simulation be used to define the requirements of the proposed prediction model?

1.5 Delimitations

This thesis investigates the feasibility of predicting Maintenance Opportunity Windows (MOWs) in a sawmill production environment through the combination of discrete-event simulation and machine learning. Several delimitations were established in order to maintain a feasible project scope.

First, the study is limited to production-system behaviour and does not consider the condition of individual machine components. Consequently, the proposed framework predicts maintenance opportunities arising from production dynamics rather than maintenance needs originating from asset degradation. The work therefore focuses on Opportunistic Maintenance rather than on Condition-Based or Predictive Maintenance applications.

Second, the machine-learning model is trained and evaluated using data generated by a simulation model. Although the methodology aims to support future industrial implementation, no real production data is used for model training or validation within the scope of this study.

Third, the evaluation of predicted Maintenance Opportunity Windows is restricted to their identification and feasibility. The study does not address the optimisation of maintenance schedules, maintenance-resource allocation, workforce planning, or spare-parts management. Similarly, no maintenance actions are automatically scheduled or executed based on the identified opportunities.

Fourth, the simulation model focuses exclusively on the production process within the sawmill. Internal transportation activities, vehicle routing, and logistics operations are excluded from the model. Likewise, the simulation does not aim to represent the physical layout of a specific industrial site in detail.

Fifth, the chosen ML to apply will be an LSTM model, to continue with the line of work from Bokrantz et al. (2025), validating their approach in a different context and refining the outcome of their model.

Finally, the study is limited to the sawmill industry and to the production assets represented in the developed simulation model. Although the proposed methodology may be applicable to other manufacturing environments, its generalisation beyond the analysed context is not explicitly evaluated.

2
Frame of Reference

2.1 Maintenance Perspective

2.1.1 Maintenance Strategies

Maintenance strategies can be classified according to the extent to which they anticipate failures and support decision-making. The most basic strategy is Corrective Maintenance (CM), which consists of restoring equipment functionality after a failure has occurred. Although simple to implement, this approach may lead to unplanned downtime, production losses, and increased maintenance costs.

Preventive Maintenance (PM) seeks to reduce the occurrence of failures by performing maintenance activities at predefined intervals based on time or usage. While this approach generally improves reliability when compared with CM, interventions may still be performed unnecessarily if the equipment remains in good condition.

Condition-Based Maintenance (CBM) extends this concept by relying on indicators of equipment condition. Maintenance activities are triggered according to observed degradation rather than fixed schedules, allowing interventions to be better aligned with the actual state of the equipment.

Predictive Maintenance (PdM) further develops this approach by utilising historical and real-time data together with analytical or machine-learning models to predict future failures and estimate the remaining useful life of assets. This enables maintenance activities to be planned before failures occur while minimising unnecessary interventions.

Finally, Prescriptive Maintenance builds upon predictive capabilities by recommending specific maintenance actions and schedules. Such systems may consider operational constraints, production requirements, and maintenance resources when determining the most appropriate course of action. Prescriptive maintenance is usually a combination of knowledge-based and data-driven decisions, and presents self-learning and adaptive capabilities integrated via feedback loops (Giacotto et al., 2025).

2.1.2 Importance of Maintenance

Maintenance plays a key role in a company's operations as well as in its financial performance. Thomas & Weiss (2021) show that, although maintenance-related metrics vary across countries, maintenance costs can still be estimated within a range of 15% to 70% of the value of produced goods.

Historically, maintenance has often been perceived as a necessary evil, as its impact on financial accounts is typically viewed as purely negative—an expense incurred to avoid greater losses. Oliveira & Lopes (2019) introduce a maturity model in which organisations with low maintenance maturity regard maintenance as unavoidable yet undesirable. Companies that recognise the importance of PM are considered slightly more mature. In contrast, highly mature organisations treat maintenance as a strategic activity, essential for achieving organisational objectives and improving quality, productivity, and cost efficiency.

Through three case studies, Salonen & Bengtsson (2011) demonstrate the benefits of aligning maintenance policies with organisational goals. These benefits include reduced unplanned downtime, increased availability, lower maintenance-related downtime, and more stable production. Such improvements contribute to reducing what Thomas & Weiss (2021) define as "lost sold value," referring to production not achieved due to machine unavailability. This highlights how maintenance can act as a cost-reduction mechanism rather than merely an unavoidable expense.

The adoption of proactive and more sophisticated maintenance policies has increased over time, as discussed by Garg & Deshmukh (2006) and Pintelon & Parodi-Herz (2008). Although such policies may initially be perceived as costlier than reactive approaches, they can ultimately improve overall company performance, as shown by Pintelon et al. (2006).

2.2 Maintenance in the Sawmill Industry

Compared with many other manufacturing sectors, the scientific literature addressing maintenance practices in the sawmill industry remains limited. While extensive research exists on maintenance management, predictive maintenance, and asset management in manufacturing environments, relatively few studies focus specifically on sawmills and wood-processing facilities.

Industrial observations and interviews conducted throughout this project suggest that maintenance practices within the sector remain largely traditional. CM and PM performed during planned production shutdowns continue to represent the dominant approaches. The adoption of more advanced maintenance strategies, such as Condition-Based Maintenance, Predictive Maintenance, or data-driven maintenance planning, appears to be limited.

Several factors may contribute to this situation. Sawmills often rely on legacy equipment, limited data-collection infrastructure, and maintenance practices that have evolved through operational experience rather than through systematic data analysis. Furthermore, the limited availability of sector-specific research may reduce the transfer of advanced maintenance methodologies from academia to industrial practice.

This lack of research presents both a challenge and an opportunity. On one hand, the scarcity of scientific studies limits the availability of validated approaches specifically tailored to sawmill production systems. On the other hand, it creates significant potential for research contributions and industrial improvements through the adaptation of maintenance concepts that have already proven successful in other manufacturing sectors.

Consequently, the remainder of this frame of reference draws primarily on the broader manufacturing and maintenance literature, while evaluating the applicability of these concepts within a sawmill production context.

2.3 Opportunistic Maintenance

Among the different approaches to maintenance policy improvement, Opportunistic Maintenance (OM) has gained increasing attention. Ab-Samat & Kamaruddin (2014) identify a growing research interest in OM, although industrial applications remain limited. Since 2000, this trend has accelerated, with several consistent benefits reported: (I) reduction of unexpected failures, (II) lower maintenance costs through the combination of PM and CM, reducing setup and downtime costs, (III) increased reliability, and (IV) improved production performance and stability.

Traditionally, OM has been viewed from a component-oriented perspective. Ab-Samat & Kamaruddin (2014) describe OM as the practice of taking advantage of existing maintenance interventions to perform additional maintenance activities. Typical examples include replacing components that are approaching the end of their useful life during a scheduled PM intervention, or combining multiple maintenance tasks into a single shutdown. Bokrantz et al. (2025) refer to this perspective as the *Component View* of Opportunistic Maintenance. The component view can be structured in three categories, as suggested by Nicolai & Dekker (2008) and Vu et al. (2020). These categories are defined based on the dependence that triggers the opportunity, and are listed: (I) Economic dependence, which relates to economies of scale when performing maintenance operations; this is, grouping maintenance activities could result in a lower cost that doing them separately. (II) Stochastic dependence, which concerns the potential increase on failure probability for certain components based on the random failure of others, and suggests that the failure of the one is then an opportunity to perform maintenance on the others as well as they might have been affected. (III) Structural dependence, which relates to the necessity of removing certain components to maintain others; because of this, the opportunity consists in performing maintenance in the removed components as well.

However, Bokrantz et al. (2025) identify an alternative perspective, referred to as the *Flow View* of Opportunistic Maintenance. Rather than focusing on the condition of individual components, the Flow View considers the behaviour of the

production system as a whole. The central idea is that certain periods arise during production in which maintenance activities can be performed without negatively affecting system throughput.

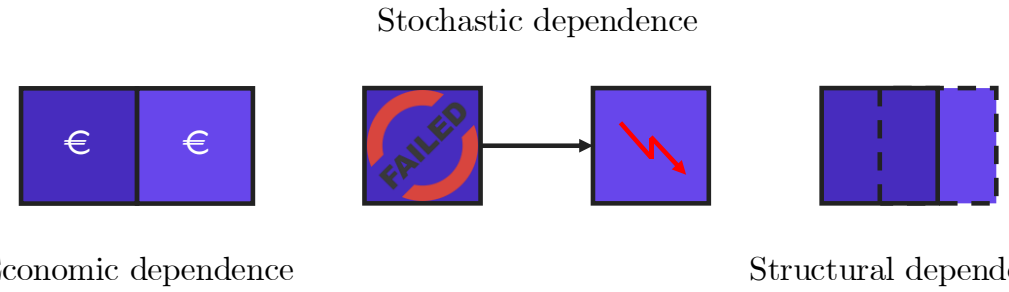

Figure 1 The three component dependences, as defined by Nicolai & Dekker (2008) and Vu et al. (2020).

These periods are defined as *Maintenance Opportunity Windows* (MOWs). A MOW can be described as a period during which a production asset is temporarily unable to generate value due to system conditions, making it available for maintenance without causing additional production losses. The exploitation of these windows may occur, for example, by (I) temporarily stopping specific production units while others continue operating using buffer capacity, or (II) performing maintenance during periods when machines are starved or blocked downstream.

The identification and exploitation of MOWs enables maintenance activities to be integrated more closely with production operations. Consequently, maintenance interventions can be performed with a reduced impact on throughput, creating opportunities to improve equipment condition while minimising production disruptions.

2.4 Data-Driven Bottleneck Analysis

2.4.1 Bottlenecks in Production Systems

The concept of bottlenecks is closely related to the objectives of this thesis. Maintenance Opportunity Windows emerge from the dynamic behaviour of production systems, particularly from situations where machines become starved or blocked due to the interactions between production assets and buffers. These phenomena are often directly associated with bottlenecks. Consequently, understanding how bottlenecks arise, evolve, and can be identified provides an important theoretical foundation for the prediction of Maintenance Opportunity Windows.

A bottleneck can be broadly defined as the resource that limits the throughput of a production system. Roser et al. (2001) define bottlenecks as resources that constrain the overall performance of the system, such that increasing their capacity would result in an increase in total system throughput.

Since production systems are composed of interconnected resources, the behaviour of bottlenecks has a significant influence on the performance of upstream and downstream assets.

Traditionally, bottlenecks were often regarded as static resources, meaning that the same machine or workstation was assumed to constrain production over extended periods. Early research highlights the decoupling effect of buffers. For example, Gershwin & Schor (2000) discuss the monotonicity property of production lines, whereby increasing buffer capacity increases throughput. This effect can be attributed to the ability of buffers to absorb variability and reduce the occurrence of both upstream starvation and downstream blockage. Still, variability in processing times, machine failures, planned stops, product mix, and buffer levels can cause the system constraint to move between different resources over time. Roser et al. (2001) describe this phenomenon as shifting bottlenecks, where different resources become the dominant constraint during different periods of operation.

The distinction between static and shifting bottlenecks is important because bottlenecks directly influence system throughput. Throughput represents the rate at which a production system generates finished products. When a bottleneck experiences downtime, reduced performance, or increased variability, the throughput of the entire system may be negatively affected. Similarly, bottlenecks may create starvation and blockage effects in neighbouring resources, causing machines to remain inactive despite being technically available for production. These inactive periods are particularly relevant in the context of Opportunistic Maintenance, as they may constitute Maintenance Opportunity Windows.

Historically, bottleneck identification relied heavily on manual observations and engineering expertise. However, the increasing availability of production data has enabled the development of data-driven approaches capable of detecting, analysing, and even predicting bottlenecks automatically. Such methods make use of machine states, production events, utilisation levels, and other operational data to identify system constraints objectively and continuously. These developments have created new opportunities to understand production-system dynamics and support operational decision-making using real-time information (Subramaniyan et al., 2016).

The following subsections review existing research on data-driven bottleneck analysis, focusing on methods for bottleneck detection, prediction, and diagnosis. Particular attention is given to the work of Subramaniyan et al., whose research demonstrates how production-state data can be utilised to identify and predict system constraints. These concepts are closely related to the prediction of Maintenance Opportunity Windows, as both rely on understanding and forecasting the dynamic behaviour of production systems.

2.4.2 Data-Driven Bottleneck Detection

Subramaniyan et al. (2016) observe that traditional bottleneck detection approaches often relied on the development and experimentation of simulation models. Although effective, such approaches typically require considerable time and effort to develop, validate, and analyse. The authors further note that recent advances in data collection technologies have enabled alternative approaches that utilise production data directly for bottleneck identification.

The proposed method builds upon the shifting bottleneck detection approach developed by Roser et al. (2002) that can be seen in figure 2. This method relies on the concept of machine activity. A machine is considered *active* whenever it is either performing an operation, such as processing material, or undergoing an activity that directly affects its operation, such as a repair following a failure; this can be seen in figure 3. The machine exhibiting the longest uninterrupted active period at a given time is defined as the current bottleneck. When the active period of this machine overlaps with the active period of another machine, the bottleneck may shift from one resource to another, giving rise to the concept of a shifting bottleneck.

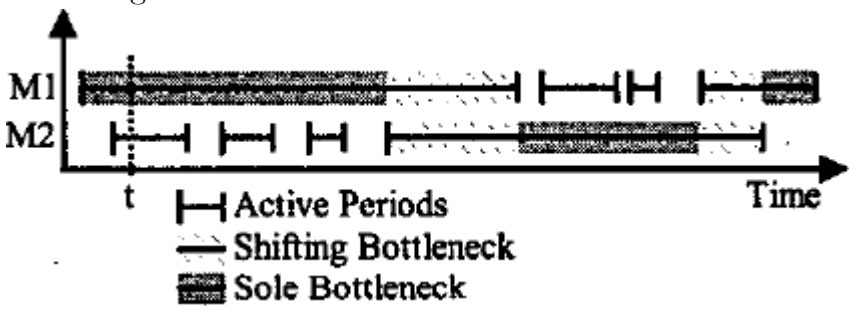

Figure 2 Representation of shifting bottlenecks by Roser et al. (2002).

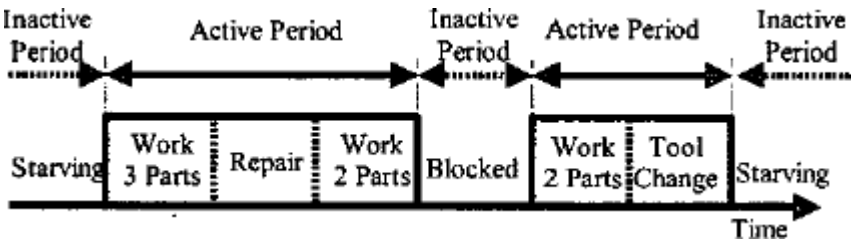

Figure 3 Representation of active and inactive periods by Roser et al. (2002).

The existence of shifting bottlenecks reflects the dynamic nature of modern production systems. Variations in machine availability, processing times, failures, and buffer levels may cause different resources to become throughput constraints at different moments in time. Consequently, bottleneck analysis must be treated as a dynamic problem rather than a static one.

The main contribution of Subramaniyan et al. (2016) is the successful adaptation of the shifting bottleneck methodology to a fully data-driven context. Rather than relying on simulation models, the proposed approach identifies bottlenecks directly from production data. This significantly reduces the effort

required for bottleneck analysis while enabling continuous monitoring of production systems using real operational data. Since bottlenecks directly influence system throughput, the proposed methodology also provides a foundation for throughput-oriented production improvement and for subsequent developments in bottleneck prediction and diagnosis.

2.4.3 Throughput Bottleneck Analysis and Prediction

A later paper by Subramaniyan et al. (2018b) expands upon the active-period bottleneck detection methodology by demonstrating that active periods are not only useful for identifying bottlenecks, but also for diagnosing why a machine behaves as a bottleneck. The authors argue that many traditional bottleneck detection methods successfully identify constrained resources but provide limited support for understanding the underlying causes of bottleneck behaviour. By aggregating machine states into active periods and subsequently decomposing those active periods into their constituent states, the proposed approach generates diagnostic insights that support maintenance and production improvement decisions. This enables engineers not only to identify bottlenecks, but also to understand whether they originate from factors such as failures, setup activities, or other operational conditions.

Subramaniyan et al (2018a) further extend the active-period concept towards the prediction of future bottlenecks. The proposed method forecasts machine active periods using autoregressive time-series models and subsequently infers future bottlenecks by statistically comparing the predicted active periods of the different machines. The study is particularly relevant because it shifts the focus from descriptive analysis of historical production behaviour towards predictive decision support.

The authors highlight that bottlenecks are dynamic in nature and may shift between resources over time due to failures, variability, and changing operating conditions. Consequently, a machine identified as a bottleneck during one production run may not remain the bottleneck during future runs. Predicting these changes enables maintenance and production resources to be allocated proactively towards the resources most likely to constrain future throughput.

An additional contribution of the study is its discussion regarding the validation of predictive models. The authors note that predictive approaches evaluated solely through simulation often report better performance metrics than those validated using industrial production data. This can be attributed to the nature of simulation environments, where system behaviour follows predefined assumptions and statistical distributions. Consequently, the paper highlights the importance of complementing simulation-based validation with real production data whenever possible, as relying exclusively on simulation results may lead to overly optimistic expectations regarding model performance in industrial applications.

Together, these studies demonstrate the evolution from data-driven bottleneck detection towards bottleneck diagnosis and prediction. More broadly, they illustrate how machine-state data can be used not only to explain historical production behaviour but also to anticipate future system conditions. This principle constitutes an important foundation for the present research, where future machine states are predicted to identify Maintenance Opportunity Windows before they occur.

2.4.4 Maintenance Perspective on Bottlenecks

Subramaniyan et al. (2020) extend previous research on data-driven bottleneck analysis by approaching throughput bottlenecks from a maintenance perspective.

The study identifies unplanned stops as one of the principal contributors to reduced availability in bottleneck resources and links the bottlenecks' existence to an availability problem. Consequently, improving the availability of bottleneck resources through appropriate maintenance actions can contribute directly to increasing system throughput.

To address this challenge, the authors propose a data-driven approach that combines bottleneck identification with maintenance diagnostics. Building upon previous active-period-based bottleneck methods, machine-learning techniques are used to analyse patterns in unplanned stop data and provide diagnostic insights into the causes of bottleneck behaviour. Rather than simply identifying which machine acts as a bottleneck, the approach aims to explain why the bottleneck emerges and which maintenance-related factors contribute most significantly to its occurrence, establishing relations between, for example, the impact of products undergoing production and the emergence of unplanned stops.

The relationship established between machine states, unplanned stops, maintenance activities, bottleneck behaviour, and throughput is particularly relevant to the present research. Since Maintenance Opportunity Windows emerge from the dynamic interactions between machine availability, starvation, and blockage conditions, understanding the maintenance-related causes behind production-system behaviour provides an important foundation for predicting future maintenance opportunities.

2.5 Prior Work on MOW Prediction

Research explicitly addressing the prediction of Maintenance Opportunity Windows (MOWs) remains limited. One of the few identified studies focusing directly on this problem is the work of Bokrantz et al. (2025), which proposes a machine-learning-based approach for predicting future MOWs from production data.

The authors build upon previous research in data-driven bottleneck analysis, particularly the work of Subramaniyan et al (2018a), which demonstrated the feasibility of predicting future machine active periods. Bokrantz et al. (2025) establish a connection between bottlenecks and the emergence of MOWs, arguing that bottlenecks are major contributors to machine starvation and blockage, which can create opportunities to perform maintenance activities without negatively affecting system throughput.

Based on this observation, the proposed methodology focuses on predicting future machine activity and inactivity. Inactive periods are interpreted as potential Maintenance Opportunity Windows. To achieve this, the authors utilise a Long Short-Term Memory (LSTM) neural network trained on historical production data. The model receives machine-state information as input and predicts future machine inactivity for fixed time windows over the prediction horizon.

The study was conducted within an automotive manufacturing context characterised by a substantially different production environment from the one considered in the present research. Furthermore, the proposed approach defines MOWs using a fixed-length windows. Any predicted inactive period matching the predefined window length is considered a MOW, regardless of the underlying cause of the inactivity. Consequently, the method does not differentiate between different types of inactivity, such as starvation, blockage, or planned stops.

The model was trained using approximately two years of historical production data collected from the industrial case study. The reported results demonstrate that production-state data can be used successfully to predict future Maintenance Opportunity Windows, providing an initial proof of concept for the application of machine-learning techniques within the field of Opportunistic Maintenance.

The work of Bokrantz et al. (2025) constitutes the primary foundation of the present study. However, several differences exist between the two contexts. Most notably, the current research investigates the applicability of MOW prediction within a sawmill environment, where production dynamics differ substantially from those of automotive manufacturing. Additionally, the present study explores the identification of feasible Maintenance Opportunity Windows beyond the simple prediction of machine inactivity, considering the practical requirements associated with performing maintenance activities within the predicted windows.

2.6 Machine Learning

This project relies on Machine Learning (ML), specifically the development of a Long Short-Term Memory (LSTM) model. Machine Learning is a field with applications across domains such as manufacturing, healthcare, and economics. It

involves the development of models and algorithms that learn from data to enable prediction or decision-making.

LSTM models are based on Recurrent Neural Networks (RNNs), inheriting their ability to capture temporal dependencies and contextual information. However, traditional RNNs suffer from vanishing and exploding gradient problems, which reduce their effectiveness over long sequences. LSTM addresses this limitation through the introduction of gating mechanisms, including a forget gate that determines which information is retained or discarded from the memory cell.

To develop Machine Learning models, Python can be considered the most widely accepted solution, both in academia and industry. Several ML frameworks are available, including PyTorch and TensorFlow. Among these, PyTorch is particularly popular in academic research and is therefore adopted in this thesis.

High-quality data is essential for developing a reliable model, as is the definition of appropriate input variables (features). To obtain the required data and identify relevant features, this study employs a simulation model. This approach mitigates the lack of existing data collection infrastructure and helps determine the necessary investments in data acquisition systems for real-world implementation.

Many projects and models using different prediction techniques can be found in the existing literature. In the field of time-series forecasting, Kolambe & Arora (2024) classify these techniques into classical methods, machine learning models, deep learning models, and hybrid approaches. Classical methods are generally based on statistical and regression techniques, while deep learning models can be distinguished from other machine learning approaches by their use of neural network architectures. Hybrid approaches combine elements from several of these categories.

Due to the variety of available models, several performance metrics are used to enable comparison. In classification problems, these metrics are commonly derived from the confusion matrix, which defines the concepts of false positives (FP), false negatives (FN), true positives (TP), and true negatives (TN), as shown in Table 1.

Table 1 Confusion matrix.

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Definitions of these metrics are widely available in the literature. Naidu et al. (2023) define Accuracy (A) as the ratio of correct predictions to all predictions (Equation 1). Precision (P) is the ratio of true positives to all predicted positives (Equation 2). Recall (R), also commonly referred to as sensitivity, is the ratio of

true positives to all actual positive cases (Equation 3). The F1-score is the harmonic mean of precision and recall (Equation 4). Since the harmonic mean penalises imbalances between precision and recall, the F1-score can be used as a combined performance metric..

$$A = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$P = \frac{TP}{TP+FP} \quad (2)$$

$$R = \frac{TP}{TP+FN} \quad (3)$$

$$F1 = 2 \cdot \frac{P \cdot R}{P+R} \quad (4)$$

Narrowing the problem definition, the present study addresses a multi-class classification problem in which mutually exclusive machine states are predicted. For this type of problem, Bishop (2006) identifies the cross-entropy loss function as a suitable objective function for model training. Cross-entropy loss is widely employed in probabilistic classification tasks and corresponds to the negative log-likelihood of the observed class labels. The loss function penalises both incorrect predictions and predictions made with low confidence, thereby encouraging the model to assign high probabilities to the correct classes. Wen & Li (2023) found that traditional methods tend to produce less accurate results and are less capable of capturing relationships between variables. They also highlight the wide applicability of Long Short-Term Memory (LSTM) algorithms for time-series data. Li & Li (2025) compare machine learning and deep learning models and show that deep learning models perform better across the evaluated metrics, including accuracy, precision, recall, and F1-score. Among the evaluated deep learning techniques, LSTM showed the lowest performance; however, the difference compared with other deep learning models was minor relative to the difference between deep learning and conventional machine learning models. Since LSTM is the simplest of the evaluated deep learning models and its performance remains close to that of more complex alternatives, it represents a cost-effective candidate for prototyping the proposed prediction model and evaluating the feasibility of applying these techniques in a sawmill production context.

LSTM belongs to the family of Recurrent Neural Networks (RNNs). RNNs are designed to process sequential data by retaining information from previous time steps, allowing temporal dependencies to be considered when generating new outputs. However, standard RNNs are affected by vanishing and exploding gradient problems, which limit their ability to learn long-term dependencies. LSTM addresses these limitations making use of gating mechanisms, including a forget gate that controls which information is retained or discarded from the cell state. As illustrated in Figure 4, a value close to zero in the forget gate reduces the influence of previous cell-state information on the updated cell state.

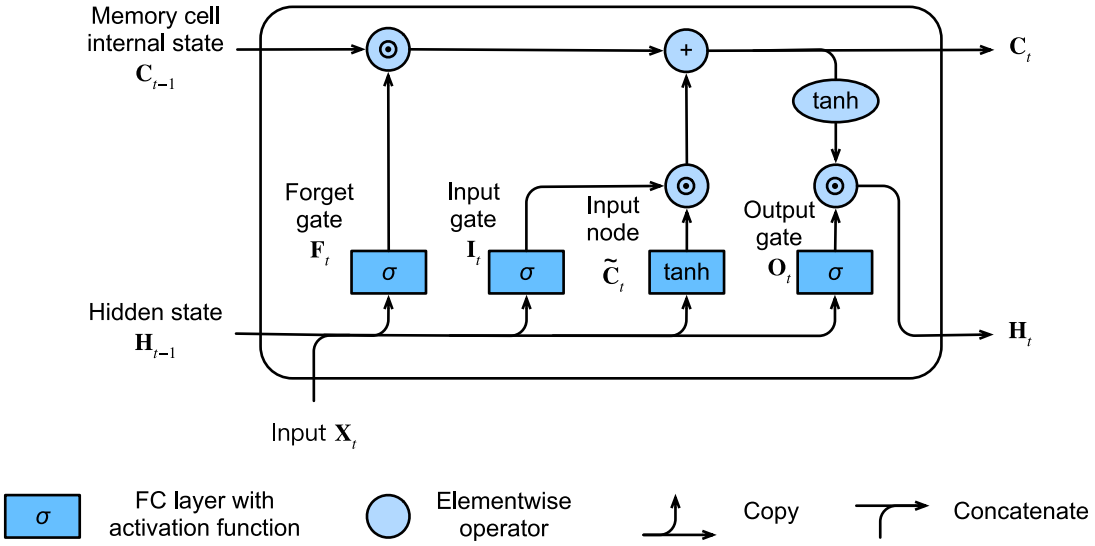

Figure 4 Architecture of LSTM (Zhang et al., 2023)

2.7 Simulation Modelling

Discrete-Event Simulation (DES) models the behaviour of a system through a sequence of events. Unlike continuous simulation approaches, where time progresses continuously, DES advances time in discrete steps, moving from one event occurrence to the next. DES models can be either deterministic or stochastic, depending on whether randomness is incorporated. In industrial systems, stochastic elements are commonly used to represent uncertainties such as machine failures.

When randomness is present in a simulation model, Monte Carlo simulations become particularly interesting. The name comes from the Monte Carlo casino in Monaco, and the method consists in running the same simulation several times, obtaining a different result each time due to the inference of random numbers and the merging the results to obtain an approximate value that can be used as a final result.

Since computers generate random numbers using deterministic algorithms, these numbers are technically pseudorandom. Pseudorandom number generators can be controlled through a parameter known as a seed, which defines the sequence of generated values. This provides a practical advantage: different scenarios can be explored by varying the seed, while reproducibility is maintained by fixing it when needed—for example, to enable fair comparisons across different model configurations.

When developing simulation models, the methodology proposed by Banks et al. (2010) provides a relevant framework. It consists of a 12-step process, ranging from problem formulation to the implementation of results. The steps are as follows:

(I) Problem formulation: defining what is to be simulated and why.

(II) Setting objectives: establishing the purpose of the simulation and defining measurable outputs that allow comparison with the current system.

(III) Model conceptualisation: identifying the elements included in the simulation model, the process flow, redundancies, delimitations, and the desired level of detail.

(IV) Data collection: gathering the data required to develop the model, such as failure rates and processing times.

(V) Model translation: translating the conceptual model into a simulation model, typically using simulation software.

(VI) Verification: checking the model for logic errors, bugs, and inconsistencies.

(VII) Validation: comparing the model's behaviour with that of the real system, where possible, and assessing whether the model represents reality sufficiently well for its intended purpose.

(VIII) Experimental design: defining how the simulation will be run, including the scenarios to be evaluated, the variables that differ between scenarios, and the number of replications per scenario.

(IX) Replications and analysis: executing the simulation, collecting the results, and analysing the outputs.

(X) Additional runs: performing further simulation runs when needed to refine the analysis.

(XI) Documentation and reporting: communicating the findings and documenting the model, including assumptions and modelling choices that may influence the results.

(XII) Implementation: applying the findings to the real system.

A diagram of the methodology is presented in Figure 5

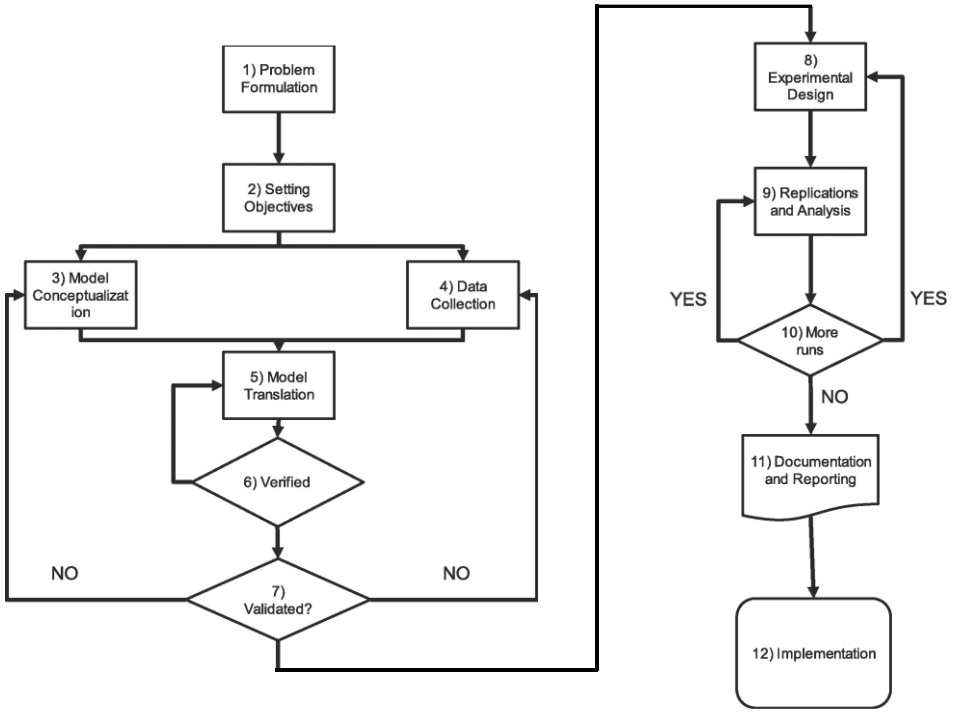

Figure 5 Simulation-based methodology process, adapted from Banks et al. (2010).

A wide range of software tools is available for DES. This study uses SimPy, a Python-based simulation framework. As an open-source library, SimPy offers a low-cost solution by eliminating licensing fees. Unlike commercial simulation software, Python does not provide built-in representations of physical production elements such as machines, conveyors, or buffers, nor considerations for 3D properties as those of materials in a conveyor or staff travelling across the production site. This results in a steeper learning curve but also offers greater flexibility, as all system logic must be explicitly defined.

This flexibility allows for the implementation of complex rules and avoids the "black-box" limitations sometimes encountered in commercial tools. Additionally, as a Python library integration with other systems becomes easier, supporting applications such as digital twins or reinforcement learning environments. SimPy provides mechanisms for event scheduling, time progression, and shared resource allocation.

3
Methodology

3.1 Definition of Industrial Case

This thesis is conducted within the framework of the READY2 project. Consequently, the industrial data, process descriptions, and operational information used throughout this work were obtained through the project and its industrial collaborators. The project provides a real-world industrial context, allowing the developed methodologies to be grounded in actual production conditions and challenges.

To preserve industrial confidentiality and facilitate the transferability of the proposed methodology, the production system is described as a generic sawmill rather than as a specific industrial facility. Nevertheless, the characteristics, processes, and parameters used in the simulation model are derived from the industrial case provided through the READY2 project. The following section describes the production system that serves as the basis for the development of the simulation and machine-learning tools presented in this thesis.

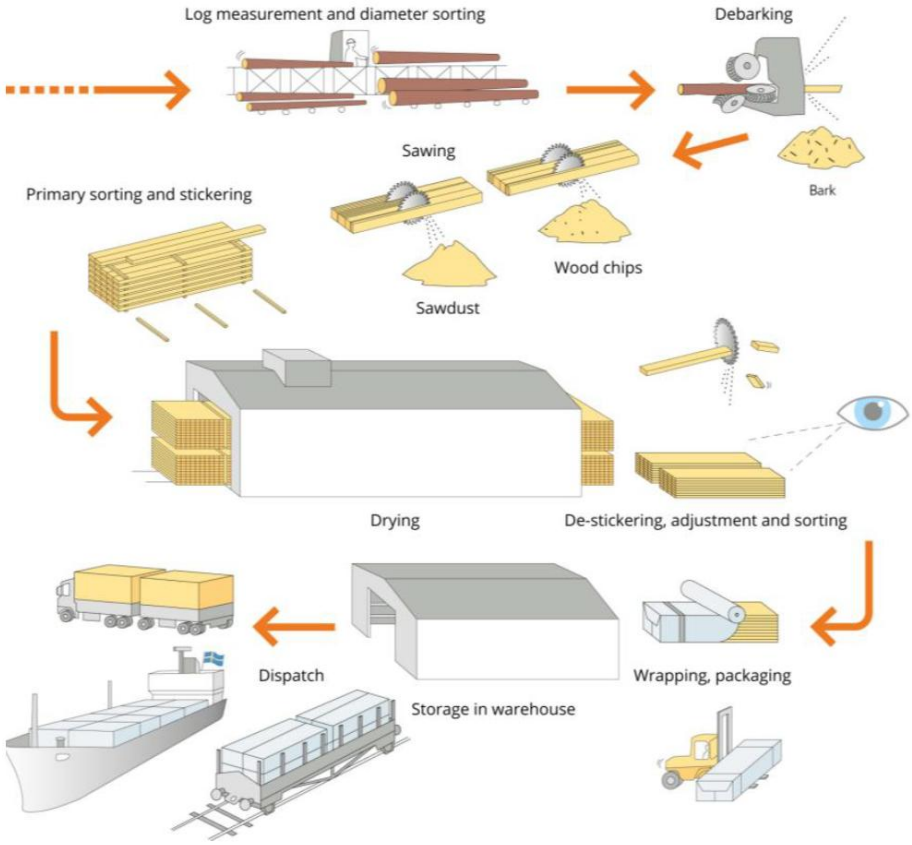

Figure 6 Overview of Sawmill process (*From Raw Material to Wood Product*, 2025).

The system begins with logs arriving to the production site by truck, after being harvested from the forest. When the timber arrives, a metal detector will check if there are traces of metal within the wood that could damage the sawmill's machinery. After so, timber will be sorted within the sorting conveyor. The classification of the timber depends on (I) the type of tree where the log comes from; (II) the diameter of the log and (III) the length of the log. The codification of categories will use the following pattern: depending on the tree species, a letter will be assigned; The diameter will fall within a range, and a number will be given based on this range. Finally, S ("Short") or L ("Long") will be considered based on the log's length. When a log is sorted, it will travel a certain distance across the conveyor. When this distance is travelled, the log will be pushed to a side compartment dedicated to logs of its class.

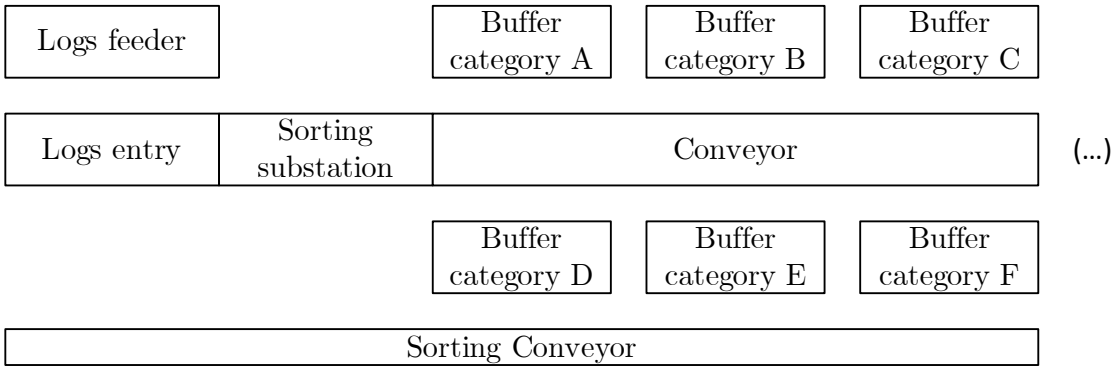

Figure 7 Representation of a logs sorting conveyor.

From there, logs are transported towards larger timber stacks, that are also class exclusive. It is of utmost importance noting that logs dry naturally, and this can impact the quality of the final product; similarly, other natural threats such as insects or fungi can impact quality to. Hence, logs cannot be left indefinitely in the stacks waiting to be processed. Moreover, it is important to note that when a timber stack begins to be transported towards the next production step, its contents will follow a Last-in-first-out (LIFO) basis, involving that the oldest logs is accessed the last. Because of this, when a stack begins to be processed, the stack must be depleted before moving onto the contents of another one.

The next step is the sawing step where logs adopt a form closer to that of a finished product, such as planks. Just before sawing, logs will be debarked. The location of this process in the process chain might vary from site to site, but it will be always performed before the sawing step. In our case, the sawing operation takes place in a conveyor-alike machine, where logs go through and eventually meet sawing blades, positioned in a specific pattern to obtain the desired cuts from the log. How blades are positioned will depend on the class of logs that is currently being sawn and will define the dimension of obtained planks; this sawing patterns are meant to maximise the obtention of material from the logs, and in some cases might produce planks of different dimensions from the same log (see

Figure 8). Besides planks, other subproducts can be obtained from the sawing step, such as wood chips or bark.

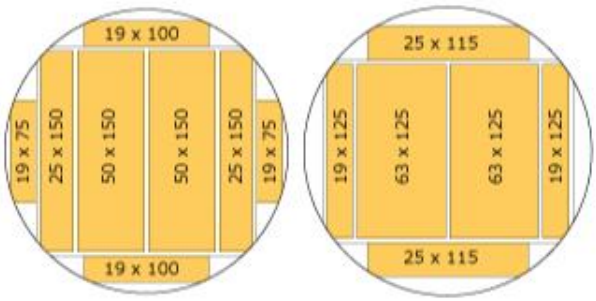

Figure 8 Example of different sawing patterns for a given log class (Linnaeus University, 2025)

Once logs are sawn and they exit the sawing conveyor, the planks are directed towards a machine that will stack them to conform a pallet. To be stacked in a pallet with another planks, the planks must have the same cross dimension, delimited by the sawing pattern, and belong to the same species of tree. The length, however, is indifferent. This is due to the to the cross dimension and tree species defining the required drying time for the planks. Inappropriate drying times for a machine would affect the quality of sawn wood. It is important to note that the drying process itself can continue outside normal working hours; however, dryers may only be loaded and unloaded during working hours. There are two types of dryers: static dryers, where all material is inputted and outputted simultaneously; and continuous dryers, where material is introduced and extracted regularly, imitating the flow concept of a conveyor.

Once the sawn material has completed the drying process, it progresses to the final stage of production. The planks are first depalletised and subsequently enter the inspection conveyor. As its name suggests, this station consists of a conveyor where planks are visually inspected by a machine which is supervised by an operator. However, the station also fulfils an important sorting function.

For example, as discussed previously, planks of different lengths belonging to the same product category may coexist within a single pallet. The inspection process separates these products into their corresponding categories. Similarly, products of different quality grades may belong to the same product family. Since different quality grades satisfy different customer requirements and command different market values, accurate classification is essential.

Sorting is achieved by directing planks into different buffers located at the end of the inspection conveyor. Once a buffer reaches a predefined occupancy level, its contents are released, provided there is no risk of mixing different product categories in downstream operations.

After leaving the sorting buffers, the planks enter the final processing stage. Release from the buffers typically occurs when the desired package size has been reached. The planks are then transported through the planer conveyor, where surface treatment is performed using planer equipment. Finally, the packages are

wrapped and labelled by operators before being transported to the finished-goods warehouse.

3.2 Simulation

The first step is to develop a simulation model of the selected sawmill in order to generate production data suitable for training the LSTM model.

A simulation model of the targeted sawmill already exists in Siemens Plant Simulation. However, this model is overly simplistic and does not capture many of the features and specificities of the real system, limiting its representativeness. Furthermore, although Plant Simulation can be more intuitive during early development stages, it becomes less flexible when modelling complex system behaviours, as discussed in the Theoretical Background section.

For these reasons, a new model will be developed from scratch following the methodology proposed by Banks et al. (2010). To support model development and gain insight into the production system, the following sources of information will be used: (I) internal documentation provided by the sawmill company, originally developed for educational purposes in simulation-related courses; (II) weekly meetings with a company representative, with the possibility of consulting additional personnel when required; and (III) a field visit to the production site to better understand the operational environment and production flow. Model validation will be performed by comparing simulated production outputs with real production data.

One of the objectives is to use the simulation model to analyse the sensitivity of the production system to buffer sizes and raw material arrival rates. This requires the design of experiments (step 8 of Banks et al.'s methodology). All relevant combinations of buffer sizes and arrival rates will be explored. Due to the stochastic nature of the system, each scenario will be simulated multiple times, and the average throughput will be used as the primary performance metric.

To manage computational cost, the experimental space will be defined with consideration of both the total number of simulation runs and the execution time per run. Fair comparisons across scenarios will be ensured by controlling the random number generation process. Specifically, the iteration index will be used to define the seed, such that the same iteration across different scenarios uses the same sequence of random numbers, while different iterations use different sequences.

In addition, each simulation component that relies on randomness will be assigned an independent random number generator. This ensures that dependencies between random variables do not alter the underlying sequences, allowing consistent comparisons across scenarios. For example, this approach is

relevant when modelling failure times that depend on machine operating conditions.

3.2.1 Available data and Parameter Identification

Most of the data used in the simulation model was obtained from official company documentation and interviews with the company supervisor. This includes the general layout of a sawmill, the existence and arrangement of machinery, conveyor characteristics such as speed, length, and spacing between items, the different log categories and their relationship to the produced plank types, pallet capacities based on plank category, drying times, and planned stops. Some parameters required site visits and interviews with specialised personnel. This was the case for category-dependent travel distances within the sorting conveyor, as well as for several intermediate buffer capacities between production assets.

The parameters governing log arrivals were not directly available. Instead, existing sorting records were analysed to estimate both the arrival frequency of logs and the quantity arriving during each event. The resulting estimates were subsequently validated through discussions with company personnel.

The treatment of random stops followed a similar approach. Historical records containing the frequency and duration of different stop causes were available in spreadsheet format. These records classify stops according to their underlying causes, not all of which correspond to equipment failures. Nevertheless, failure-related terminology is used throughout the simulation framework for consistency.

To estimate the distributions governing random stops, causes that should emerge naturally from the simulation logic—such as material starvation—were excluded from the analysis. Using the number of occurrences of each stop category and the total recorded operating time, the Mean Time Between Failures (MTBF) was calculated. Similarly, the Mean Time To Repair (MTTR) was obtained from the number of occurrences and the total accumulated duration of each stop category. To reduce implementation complexity, individual causes were aggregated into broader categories using weighted averages. The resulting MTBF and MTTR values were then used to parameterise exponential distributions representing the occurrence and duration of random stops.

3.2.2 Model delimitations, simplifications and assumptions

This section describes the principal delimitations, simplifications, and assumptions adopted during the development of the simulation model.

Beginning with log sorting, the possibility of the sorting conveyor incorrectly classifying a log and directing it to a reprocessing compartment is not modelled. The observed occurrence rate of such events is sufficiently low that their exclusion is not expected to significantly influence the results.

Although some log categories can be processed using multiple sawing patterns, the model assumes a one-to-one relationship between log categories and sawing patterns. For each log category, the retained sawing pattern corresponds to the one most frequently used in practice. This simplification is considered reasonable because, although multiple sawing patterns may be technically feasible, production records indicate that a single pattern is often overwhelmingly dominant.

Furthermore, each sawing pattern is assumed to produce a single plank category. Secondary products such as bark, chips, and pulp are not represented in the simulation model.

The drying process will consist only of static dryers to maintain development simplicity. Any continuous dryer will be modelled as a static one.

Similarly, plank categories are not differentiated beyond the distinction between short and long products during the inspection stage. Variations related to quality grades and intended end-use applications are therefore omitted, as they do not contribute to the objectives of the study.

The final packaging operations performed after the planer conveyor are also excluded. These activities are largely manual and were observed not to constrain upstream production assets. Consequently, their inclusion would increase model complexity without providing meaningful benefits in relation to the research objectives.

Finally, internal transportation is not represented in the simulation model. Preliminary interviews suggested that transportation activities may influence system performance to some extent. However, given the scope and time constraints of the project, transportation was excluded in order to limit complexity and avoid the substantial effort required to accurately represent facility geometry, travel distances, and vehicle movements within the simulation environment.

3.2.3 Definition of generic simulation assets

The simulation model was developed using SimPy. Unlike commercial simulation software, SimPy does not provide predefined assets representing material-flow elements such as machines, conveyors, or the goods being processed. Instead, it provides mechanisms for queue management and simulation-clock progression through event scheduling. As discussed previously, this offers greater flexibility than commercial software but requires the modeller to define all system behaviour from scratch.

The model was developed using an inheritance-based architecture resembling the design of a Python library. The objective of this approach is to maximise code reuse by ensuring that common attributes and methods are shared whenever possible. Consequently, all simulation assets ultimately derive from a common base class and progressively specialise through inheritance, resulting in distinct concrete elements with specific behaviours.

The machinery-related concepts are introduced by first defining the base classes from which more specialised assets inherit. Following the recommendations of PEP 8 by van Rossum et al. (2013), class names are written using the *CapitalisedWords* convention, whereas attributes and methods follow the *name_separated_with_underscores* convention. Throughout this section, class names, methods, attributes, and type hints are presented exactly as implemented in the source code.

Detailed descriptions of class attributes and methods are provided in Appendices A and B. Attributes are categorised as required inputs (R), optional inputs (O), or self-calculated attributes (C). In some cases, an attribute may belong to more than one category, typically when an optional input is used to initialise a value that is subsequently calculated automatically. Such cases are explained individually where relevant.

Attributes and methods whose names begin with an underscore (_) are intended for internal use. In accordance with common Python conventions, these elements should generally not be accessed or modified directly outside the class hierarchy. Unless explicitly stated otherwise, such elements can be regarded as implementation details that support the behaviour of higher-level classes and are not meant to be overridden under normal circumstances.

Simulation Environment

The simulation environment constitutes the foundation of the model. SimPy provides a class named Environment, which acts as the common context in which all events are scheduled and simulation time is managed. The Environment class is highly sophisticated and is generally not intended to be modified.

However, programming languages, including Python, are subject to numerical precision limitations when handling floating-point values. Although these issues can often be mitigated by rounding numerical operations, the mechanisms responsible for advancing simulation time remain inaccessible unless the Environment class itself is extended. Another common practice consists of selecting smaller time units to minimise the occurrence of decimal values (e.g., using seconds rather than hours).

For this reason, a custom class named MeasuredEnvironment is introduced. This class extends SimPy's Environment and incorporates mechanisms for handling numerical precision safely, while also simplifying interaction with the simulation model. For example, it allows users to define planned stops in intuitive units such as hours or minutes, which are then automatically converted to the internal time unit used by the simulation environment. The resulting class is intended to be used directly and is not expected to require further modification.

Random Number Generation

Having defined the simulation environment, the next step is to introduce the mechanisms responsible for random number generation. Randomness plays a central role in DES, allowing uncertainty to be represented through probabilistic distributions.

Although Python already provides several random-number generation libraries, different distributions often use distinct interfaces and parameter definitions. To simplify integration and standardise usage throughout the simulation framework, an abstract base class named SimpyRNG is defined. This class establishes a common interface for all random-number generators used within the model.

The SimpyRNG class defines a single abstract method and two common attributes that are inherited by all subclasses.

Several subclasses of SimpyRNG are provided. Each subclass implements its own version of gen_random, while differing primarily in the parameters required during instantiation. Since these parameters correspond directly to well-established statistical distributions, only a brief overview of the available classes is provided.

Material Flow

Having defined the random-number generation framework, the next step is to introduce the elements that physically move throughout the simulated production system. These elements are represented through the Material class, which acts as a base class for all goods flowing through the model.

The Material class is referenced extensively throughout the simulation framework, as production assets interact primarily through the transfer and processing of Material objects.

Logging Capabilities

Since the objectives of this thesis require the simulation to generate data suitable for machine-learning applications, logging capabilities constitute a fundamental component of the simulation framework.

To support this functionality, an abstract class named GenericLogger is defined. This class establishes the minimum interface required for simulation loggers and specifies how production assets interact with the logging system. The class itself does not implement any storage mechanism and instead defines placeholder methods that must be implemented by subclasses.

Production Assets

Production assets are responsible for processing and transferring Material objects throughout the simulation model. Examples include machines, conveyors, and other elements involved in the production flow.

The base class representing production assets is named GenericElement. This class is not intended to be instantiated directly. Instead, it serves as a

foundation from which more specialised production assets inherit their common behaviour. The class has been designed to maximise flexibility, allowing inherited classes to override methods whenever context-specific behaviour is required.

It is worth noting that the simulation framework uses attribute and method names containing the term failure to refer to the internal random-stop mechanism. However, in this context, these elements are not limited to mechanical failures. They may represent any randomly occurring stop cause, including equipment breakdowns, operator-related interruptions, or other unplanned events.

Having established the common behaviour shared by all production assets, the following sections introduce concrete classes that inherit from GenericElement. The first of these is Machine, which represents a production asset capable of processing either a single Material object or a group of Material objects whose processing starts and finishes simultaneously.

Similar to Machine, the class AsyncMachine allows the processing of one or more Material objects simultaneously. "Async" stands for Asynchronous which indicates that different Material objects may begin and complete processing at different times.

Conveyor extends AsyncMachine and represents a production asset capable of processing multiple Material objects simultaneously while allowing them to complete processing at different times.

Unlike a generic AsyncMachine, the processing state of one Material object directly affects the processing of the others. This is because conveyor processing represents the physical transportation of Material objects while maintaining fixed spatial relationships between them. Consequently, if one Material object is unable to advance—for example, because it has completed processing but is blocked downstream—the remaining Material objects are also prevented from advancing.

Furthermore, conveyors must account for the geometric properties of the Material objects currently being transported, as well as those attempting to enter the conveyor. Conveyor inherits from AsyncMachine rather than directly from GenericElement, since they will share the process method.

Buffers represent production assets where processing occurs instantaneously. Consequently, incoming Material objects remain stored within the buffer until the conditions required for transfer are satisfied. Such conditions may include downstream availability constraints or batching requirements.

For the simulation model to operate, Material objects must enter the system. This functionality is provided by the MaterialArrival class. Unlike other production assets, this class interprets processing time as the time elapsed between successive arrivals of Material objects.

3.2.4 Definition of specific simulation assets

Having defined the generic simulation assets, the inheritance-based approach can be continued by introducing more specialised classes that represent the behaviour of specific equipment found in a sawmill. These classes build upon

the generic definitions of Material objects and production assets introduced previously. Following the same approach as in the preceding subsection, only new and overridden attributes and methods are presented.

Material Flow

The material-flow elements defined for the sawmill simulation distinguish between three types of Material: Log, Plank, and Pallet.

Log objects represent the primary raw material entering the production system. Plank objects represent the final finished product and are produced from Log objects. Pallet objects are used to group multiple Plank objects into a single Material object while preserving the ability to recover the individual Plank objects later in the process.

Logging capabilities

To implement the abstract methods defined by GenericLogger, a concrete logging class is required. Since the simulation outputs are intended to be exported in .xlsx and .csv formats, the subclass ExcelLogger is defined.

The ExcelLogger class generates two production logs. The first is an .xlsx file intended for human interpretation and analysis. The second is a .csv file designed for machine-learning applications, where each row represents one minute of simulation time.

Production Assets

Most production assets exhibit behaviour that is specific to a particular stage of the production process and cannot be generalised across all asset types. This reinforces the value of the inheritance-based architecture, as it allows common functionality to be reused while only redefining the methods that require specialised behaviour.

The assets presented in this section follow the chronological order of the production process, beginning with the arrival of raw material and ending with finished products ready for shipment. It should be noted that not all assets used in the final model require specialised subclasses. In several cases, the generic classes introduced previously can be used directly.

The first specialised production asset is LogArrival, which inherits from MaterialArrival. As a subclass of MaterialArrival, its primary responsibility is defining how the information represented by product_mix_distribution and product_mix_descriptors is structured, together with the implementation of mix_generator.

Following arrival, logs are sorted using the SortingConveyor class. This class is a specialised version of Conveyor in which Material objects do not necessarily travel to the end of the conveyor. Instead, different log categories are diverted into different side compartments according to their classification. This requires support for multiple successors and specialised routing behaviour.

To better represent piles of logs, a specialised variant of Buffer is required. Log piles are governed both by their occupancy level and by the residence time of the oldest stored Log. These mechanisms are incorporated into the TimeStamperBuffer class, together with communication mechanisms required to interact with the MasterScheduler, which coordinates the selection of log piles for processing.

The next major production asset is SawingConveyor, which represents the sawing process. Conceptually, the asset behaves similarly to a conveyor; however, it incorporates additional mechanisms associated with the selection of sawing patterns. These patterns are controlled by the MasterScheduler, which may request setup operations to modify the current production configuration.

Furthermore, the class receives Log objects as inputs but outputs multiple Plank objects. Consequently, special handling is required to ensure that production tracking and occupancy calculations remain consistent.

Produced Plank objects must be grouped into Pallet objects before entering the drying process. Since palletisation is assumed to occur instantaneously, the implementation is based on a specialised Buffer class. The primary modification consists of receiving Plank objects as inputs while producing Pallet objects as outputs.

As auxiliary elements, one PalletBuffer is defined for each pallet category. These assets behave similarly to the previously defined TimeStamperBuffer objects in that they are coordinated by a central scheduling entity. In this case, coordination is performed by the DryerHandler, which will be introduced later.

Unlike conventional buffers, PalletBuffer objects operate collectively rather than independently. Furthermore, each buffer may supply multiple drying chambers, requiring support for multiple downstream destinations.

The drying process is represented by the Dryer class, a specialised subclass of Machine. Unlike a standard machine, processing times depend on the category of the incoming pallet. Additionally, loading and unloading activities are constrained by working schedules that resemble planned stops. To maintain flexibility, these mechanisms are implemented separately from the planned-stop system inherited from GenericElement.

Once the Plank objects contained within a Pallet have been dried, the pallets can be dismantled. Since depalletisation is assumed to occur instantaneously, the implementation is based on a specialised Buffer in which a single incoming Pallet results in the release of all contained Plank objects.

The final custom production asset is the InspectionConveyor, located downstream of the drying process. This asset routes Plank objects to multiple destination buffers and therefore requires support for multiple successors.

Despite representing a physical conveyor, the class inherits from Machine rather than Conveyor. Observations of the system indicated that modelling the asset using a conveyor-based implementation would incur a substantial

computational cost. Replacing it with a machine-based implementation reduced this burden while introducing a positive deviation of only approximately 0.019% in the number of produced Plank objects.

Complex interaction handling

The final elements of the simulation framework are the two scheduler classes introduced previously. Unlike the production assets described earlier, these classes do not inherit from any of the existing asset hierarchies due to their highly specialised behaviour. Instead, they act as coordinating entities responsible for synchronising the operation of multiple production assets simultaneously.

The first scheduler is MasterScheduler, which coordinates the different log piles represented by TimeStamperBuffer objects and the SawingConveyor. Its primary responsibility is determining which pile of logs should supply material to the sawing process, considering both age-related and occupancy-related constraints.

To minimise excessive drying of stored logs, priority is primarily determined by the age of the oldest log within each pile. Occupancy levels are used as a secondary criterion when age-based priorities are equivalent. Since log piles are processed according to a LIFO policy, the scheduler does not switch to a different pile until the currently selected pile has been completely emptied, including any new logs that arrive while the pile is being processed.

The interaction between PalletBuffer objects and Dryer objects is managed by the DryerHandler class. The scheduler generates production orders by considering the number of available pallets, the minimum loading requirements of the drying chambers and their maximum capacity. This ensures that dryers are not started before sufficient material is available while also prioritising the most suitable production orders.

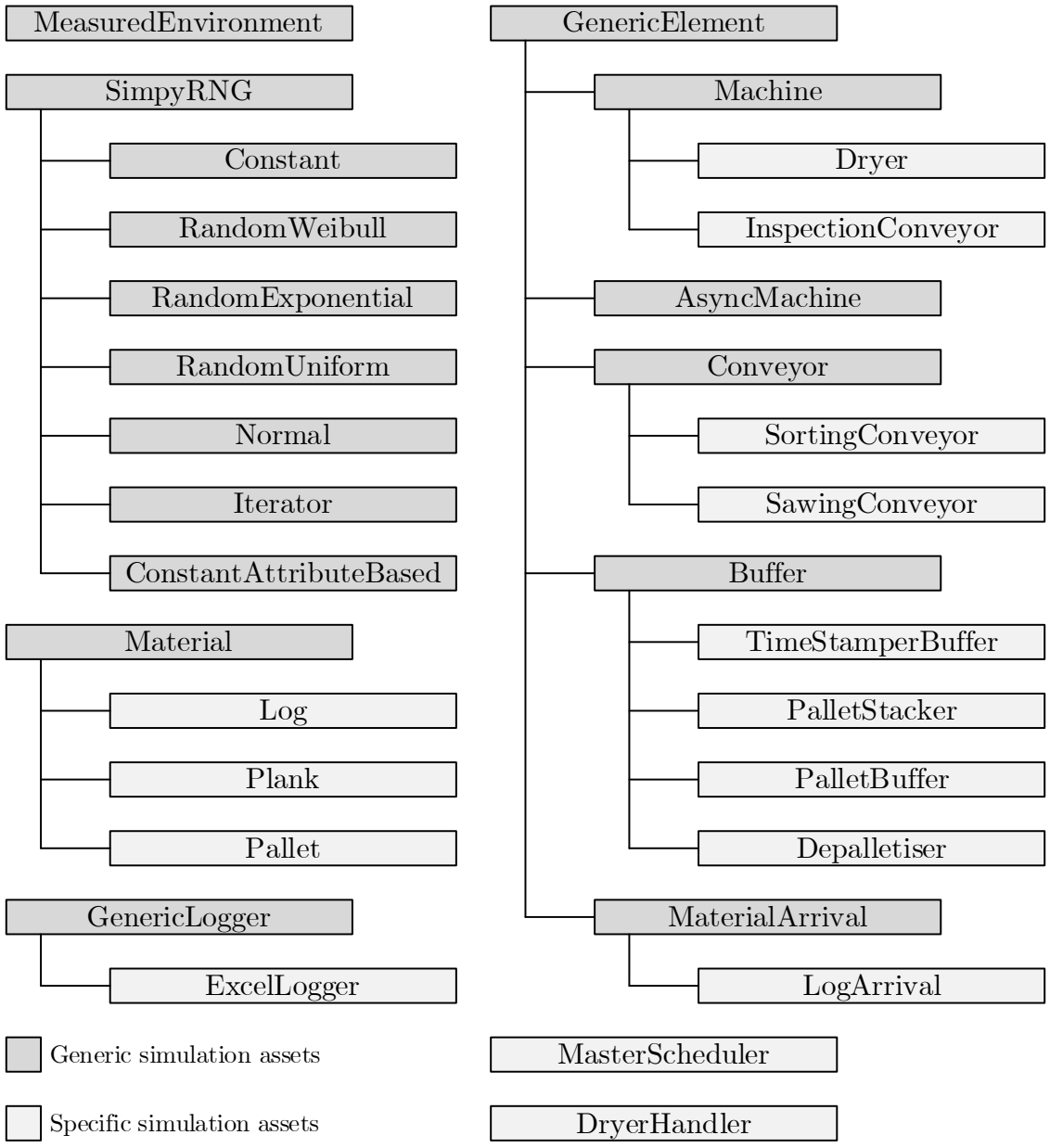

Figure 9 Hierarchy of the simulation elements.

3.3 Synthetic Data Generation

The developed simulation model is used to generate synthetic production data that can subsequently support data-driven approaches, including the development of the proposed LSTM prediction model. This approach addresses one of the principal challenges of the project: the absence of sufficient production data and the lack of an established data-collection infrastructure capable of supporting machine-learning applications.

By generating synthetic data through simulation, it becomes possible to iteratively define the data requirements of the prediction model and evaluate the

relevance of different variables before investing in real-world data-collection systems. Furthermore, simulation enables the rapid generation of large datasets that would otherwise require extended periods of industrial operation to obtain.

To facilitate data generation, a set of logging classes was developed and integrated with the simulation assets described in the previous section. These classes record production events and machine-state changes during simulation execution, producing logs that resemble those commonly found in industrial environments. The resulting event-based logs can subsequently be transformed into time-series datasets suitable for training and validating LSTM models.

3.4 Evaluation of Scenarios

Different production scenarios are evaluated using a Monte Carlo approach. The analysed scenarios are defined by combining different maximum log-pile capacities with different increases in the time between material arrival. These combinations are evaluated for each of the considered production-mix cases, resulting in the complete set of analysed scenarios.

For each scenario, multiple simulation replications are performed using different RNG seeds. To ensure fair comparisons between scenarios, the same set of seeds is reused across all scenario configurations. Consequently, the first replication of every scenario uses the same seed, the second replication uses another common seed, and so forth. This approach reduces the influence of stochastic variation when comparing scenarios, allowing the observed differences in performance to be attributed primarily to the modified scenario parameters rather than to random effects.

3.5 LSTM MOWs Prediction Algorithm

The prediction of future machine states and Maintenance Opportunity Windows requires a methodology capable of modelling temporal dependencies within production data. Machine Learning techniques, and particularly deep learning approaches for time-series analysis, provide a suitable framework for this purpose.

A clear definition of input and output variables is essential for the successful development of the LSTM model. In relation to Research Question 2, the aim is to develop a model that relies strictly on production data. This means that information related to the physical condition of assets, such as vibration, humidity, or temperature, will not be included. Instead, the model will rely on production-related variables, such as machine states, time since last repair, or buffer levels.

Initially, three additional features will be included: the net production time since the last occurrence of a Failed state, the net production time since the last

occurrence of a Setup state, both for machines where these states are possible, and the total time elapsed since the last planned stop for each machine.

The model output will consist of the predicted machine states for future time steps. The prediction horizon, meaning the number of time steps predicted ahead, will be defined later. Each time step will represent one minute. Output variables will be represented in the same way as input machine-state variables, excluding variables that do not represent machine states. The use of binary outputs formulates the task as a classification problem, which is appropriate for predicting discrete machine states. This choice is also supported by Foumani et al. (2024), who highlight the suitability of LSTM-based approaches for classification tasks in time-series contexts.

Beyond predicting machine states, the feasibility of MOWs must also be defined and detected. A MOW is understood as a time window during which a machine is in an inactive state and can therefore potentially be used for maintenance. Roser et al. (2001) define "active" machines as those being worked on, including machines that are producing, undergoing setup, or being repaired after a failure. In contrast, "inactive" machines are those waiting for another activity or event to occur, such as machines that are starved, blocked, or in a planned stop. These inactive periods may therefore represent valid opportunities for maintenance activities.

Whether a continuous inactive period can be considered a feasible MOW depends heavily on the duration of the window and the requirements of the maintenance activity. Responsible personnel within the company will be interviewed to identify existing maintenance activities, their typical duration, and any practical constraints affecting their execution.

3.5.1 Model development

The prediction algorithm will be developed in Python using the PyTorch library, which is widely adopted for machine learning development in both research and practice. To seek guidance and gain further insight into MOW prediction, interviews will be conducted with Mukund Subramaniyan, due to his expertise in the field, and Rashid Ali, due to his relevance to the machine learning development of the project.

Bringing machine learning into production can be challenging, particularly when the organisation has limited maturity in this area. Working within a structured framework can support the development process. Therefore, this study draws on the work of Krauß et al. (2023), who define key requirements for tools, methods, and techniques needed for successful machine learning projects in production contexts. Their framework, referred to as the "ML-Toolbox", is divided into three sections, as shown in Figure 10.

The first section, data interfaces, concerns the availability of high-quality data sources, the interfaces responsible for collecting data and feeding the model, and the transmission of model outputs. In this project, this role is represented by

the simulation model, which generates production data based on events occurring within the simulated system. The simulation data will be printed into .csv files to mimic the behaviour of usual data collection systems. Concerning the output, due to interfaces for visualising the final outputs falling outside the scope of the project, we will use python console and Excel as output interfaces.

The second section, data science tools, includes the tools that support machine learning activities. In this study, this includes data preprocessing, such as transforming event-based logs into fixed-time-step logs, as well as post-processing of the model output through a simple algorithm that evaluates predicted windows and determines whether they constitute feasible MOWs. PyTorch also belongs to this category, as it provides many of the methods required to develop machine learning models.

The third section, application area interfaces, concerns the boundary between automated machine learning processes and human decision-making. This includes evaluating the risks associated with automation, determining when humans should remain in the loop, and defining how users interact with the model and its outputs. Since this project is prototypical in nature, this section will be addressed primarily from a theoretical perspective and left open for further discussion.

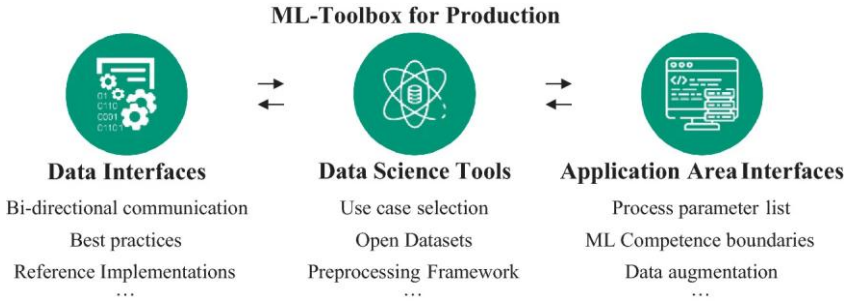

Figure 10 Requirements for a ML project, by Krauß et al. (2023).

3.5.2 Model training and evaluation

The algorithm will initially be trained using input data generated by the simulation model, which will be executed for a sufficiently long period to ensure steady-state behaviour. At present, specific data collection points have not been defined, and existing data collection systems are mainly limited to KPI tracking. However, the company has expressed willingness to invest in the required infrastructure once appropriate data requirements have been identified. Additionally, data from other facilities is available, providing insight into the expected structure and quality of real-world data.

Several production logs will be generated, with some used exclusively for training and others reserved exclusively for validation. This approach differs from the conventional train-validation split, in which a single dataset is divided into training and validation subsets. Instead, entire simulation runs are allocated to either training or validation. The objective of this strategy is to reduce the

likelihood of the LSTM model learning characteristics specific to a particular simulation run and its associated random-number sequence. Consequently, validation performance is evaluated using production patterns generated independently from those used during training and thus reducing the chances of overfitting.

A total of six production logs are generated, each corresponding to a simulation period of 120 days. Four logs (Logs 1–4) are used for training, while the remaining two logs (Logs 5 and 6) are reserved for validation.

As discussed previously, the objective function used during training is the cross-entropy loss. Since cross-entropy is based on the negative logarithm of the predicted probability assigned to the correct class, its best value is zero, whereas no finite upper bound exists. To establish a comparable baseline representing poor performance, a predictor assigning equal probability to all possible classes can be considered. Under this assumption, the cross-entropy loss is given by Equation 5, where K denotes the number of possible states.

$$CEL = -\ln\left(\frac{1}{K}\right) = \ln(K) \tag{5}$$

Since different machines may exhibit different numbers of possible states, an average baseline cross-entropy loss can be computed according to Equation 6, where K_i represents the number of states associated with machine i, and N denotes the total number of machines:

$$CEL = \frac{\sum_i \ln(K_i)}{N} = \frac{\ln\left(\prod_i K\right)}{N} = \frac{\ln(5 \cdot 6 \cdot 4 \cdot 5)}{4} = 1{,}599 \tag{6}$$

If model performance is not considered satisfactory, the selected features will be iteratively refined until acceptable results are achieved. Ideally, the number of required data collection points should be minimised to reduce implementation complexity and investment requirements.

Training and validation will be performed using the Alvis infrastructure provided by NAISS (National Academic Infrastructure for Supercomputing in Sweden). The input sequence length will consist of 10.000 time steps, corresponding to 10.000 minutes of production data. This value was selected because 10.000 minutes approximately corresponds to one week of operation, which is sufficient time for a log pile to be processed and for the resulting planks to complete the drying process.

The prediction horizon will span 90 minutes. In addition to the average cross-entropy loss used as the objective function, cross-entropy loss will also be calculated individually for each machine. Furthermore, the average cross-entropy loss will be evaluated for three separate intervals within the 90-minute prediction horizon, providing additional insight into how prediction quality evolves over time and helping assess the practical usability of the model.

3.5.3 Maintenance Opportunity Windows identification

An evaluation function will be applied to the trained model to assess its ability to identify Maintenance Opportunity Windows (MOWs). This function will execute the LSTM model at fixed time intervals using simulation logs, generate predictions of future machine states, identify potential MOWs from those predictions, and compare them with the actual system behaviour. The objective is to determine whether the predicted MOWs correspond to real opportunity windows that materialise in the production system.

To define what constitutes a feasible MOW, interviews will be conducted with company personnel to identify maintenance activities that could potentially be allocated within such windows. Particular attention will be given to the duration of these activities, allowing the definition of a minimum window length required for a time period to be considered a MOW. Additional operational constraints identified during these interviews will also be documented and discussed later in the Discussion chapter.

Once these criteria have been established, the evaluation function will be applied to the model outputs. A predicted MOW will be considered correct if it satisfies the predefined feasibility criteria and corresponds to an actual MOW observed in the production system. The reliability of the predictions will be measured as the ratio between correctly predicted MOWs and the total number of predicted MOWs, as shown in Equation 7.

$$MOW\ reliability = \frac{Correctly\ predicted\ MOWs}{Total\ predicted\ MOWs} \quad (7)$$

This metric provides a practical measure of the usefulness of the proposed prediction framework from a maintenance-planning perspective, as it evaluates the proportion of predicted opportunities that can effectively be exploited in practice.

4
Results

4.1 Simulation Model

The Monte Carlo analysis yielded several insights into the behaviour of the production system. Three cases were analysed: a generic case in which different types of logs and planks can be processed, a case in which only the log type associated with the shortest sawing time and the longest drying time is processed, and the opposite case, in which only the log type associated with the longest sawing time and the shortest drying time is processed.

For all three cases, different combinations of log piles capacities and increases in the inter-arrival time between log deliveries were evaluated.

Under the initial arrival rates and buffer capacities, the sawing equipment was identified as the clear bottleneck in all scenarios, while the dryers exhibited minor bottleneck-like behaviour. As the inter-arrival time between log deliveries increased, the influence of buffer capacity reductions became progressively less significant. This behaviour is consistent with the decoupling effect of buffers deducted from Gershwin & Schor (2000). As material shortages become the dominant constraint, variability associated with the original bottlenecks becomes increasingly irrelevant. Across all analysed scenarios, throughput proved to be considerably more sensitive to changes in material arrival rates than to changes in buffer capacities.

In the two scenarios where only a single log variant was processed, buffer capacities exhibited virtually no influence on system throughput. The observed differences between scenarios ranged from 0% to 0,5% and are considered negligible. These variations are likely attributable to stochastic variation associated with random-number generation. It should also be noted that one of the analysed log variants produces two planks per log, whereas the other produces three. Consequently, although the scenario associated with the longer sawing time processed up to 100,000 fewer logs, it still produced approximately 23,000 more planks than the other single-variant scenario.

The generic case exhibited a greater sensitivity to reductions in buffer capacity, although this effect also diminished as the inter-arrival time between log deliveries increased. Without any increase in material inter-arrival times, reducing buffer capacities from 100% to 30% of their current values resulted in only marginal differences in throughput (approximately 1%, with the best value obtained when buffers' capacity was 50% of the current maximum). These

differences are again considered primarily attributable to stochastic variation, but may also be influenced by changes in the production mix, particularly the faster transition to products generating three versus two planks per log. However, when buffer capacities were reduced to only 10% of their current values, total throughput decreased by approximately 8%. A major contributor to this reduction appears to be the increased frequency of setup operations at the bottleneck resource, caused by the more rapid transitions between log types.

4.2 Generated Simulation Data

The simulation model proved capable of generating sufficient quantities of data for machine-learning applications. The speed of data generation is primarily influenced by the available computational resources, the complexity of the simulation model, and the implemented logging mechanisms. Nevertheless, two years of production data, divided into six independent datasets, were generated in around 6-8 hours using consumer-grade hardware.

The generated logs consist of spreadsheet files containing one worksheet per machine, where state transitions are recorded together with their corresponding timestamps and durations. Although this event-based format is easily interpretable by human users, it is later transformed into a time-series representation suitable for LSTM development. In this second log, the information from all machines is consolidated into a single comma-separated values worksheet with a one-minute sampling interval. Each machine state is represented as an individual variable, resulting in a structured time-series dataset describing the behaviour of the production system over time.

In addition to the machine states, several derived variables are included to enrich the dataset. These variables comprise the net production time since the last random stop (for machines where random stops may occur), the net production time since the last setup operation (for machines where setups are applicable), and the total time elapsed since the last planned stop. The possible machine states that can be recorded are: (I) Production, (II) Planned Stop, (III) Random Stop, (IV) Setup, (V) Starvation, and (VI) Downstream Blockage.

4.3 MOWs Prediction Model

Initially, the model was trained using a conventional train-validation split. However, this approach was quickly found to be inadequate. Although the validation results appeared highly promising, evaluation on a completely independent simulation log produced substantially worse results, with performance approaching the baseline value previously defined for poor predictions. This behaviour indicated a strong tendency towards overfitting. Consequently, the training methodology was modified to the approach described

in the methodology chapter, where the validation logs are generated independently from the logs used for training.

The tendency towards overfitting can be observed in Figure 11, particularly during the early stages of training, where reductions in training loss were often accompanied by increases in validation loss.

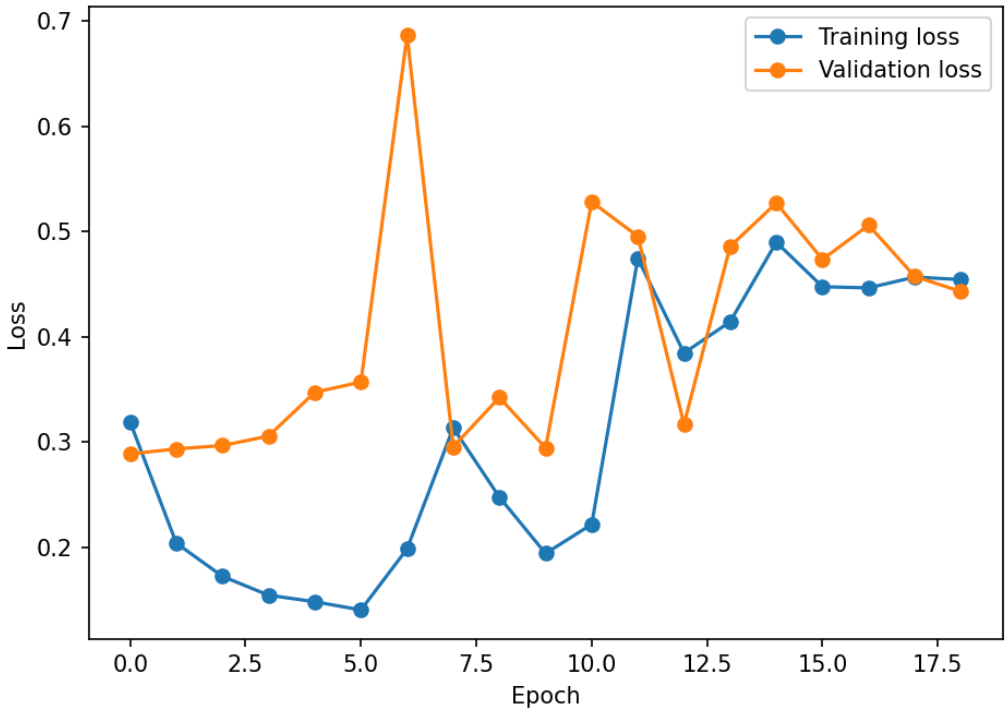

Figure 11 Training loss vs validation loss over epochs.

To further evaluate model performance, additional metrics were analysed for the best-performing model. These metrics include machine-specific performance measures and metrics related to different sections of the prediction horizon.

Machine-specific metrics provide insight into the predictive performance achieved for each production asset and are presented in Table 2. The results indicate relatively consistent performance across all machines, with the exception of the Sorting Conveyor, which exhibits a noticeably higher cross-entropy loss. Despite this difference, the accuracy values remain relatively similar across all machines, suggesting that the overall reliability of the predictions is comparable throughout the production system.

Table 2 Per-machine metrics.

	Cross-entropy loss	Accuracy
Sorting Conveyor	0,408	86,554 %
Sawing Equipment	0,245	92,037 %
Inspection Conveyor	0,253	89,309 %
Planer	0,251	91,669 %

Horizon-based metrics provide insight into how prediction quality varies as the forecast moves further into the future. For both evaluated metrics, the best performance is achieved at the beginning of the prediction horizon, while the worst performance is observed at its end. A gradual deterioration in predictive quality can be observed as the forecast horizon increases. This trend is illustrated in Figures 9 and 10.

Table 3 summarises the obtained results. In addition to the metrics corresponding to the first and last forecasted minute, the table presents average values for three equally sized sections of the 90-minute prediction horizon: the early horizon (minutes 1–30), the middle horizon (minutes 31–60), and the late horizon (minutes 61–90).

Table 3 Horizon metrics.

	Cross-entropy loss	Accuracy
1st minute	0,241	91,944 %
Early horizon	0,263	91,035 %
Middle horizon	0,292	89,703 %
Late horizon	0,312	88,939 %
90th minute	0,319	88,595 %

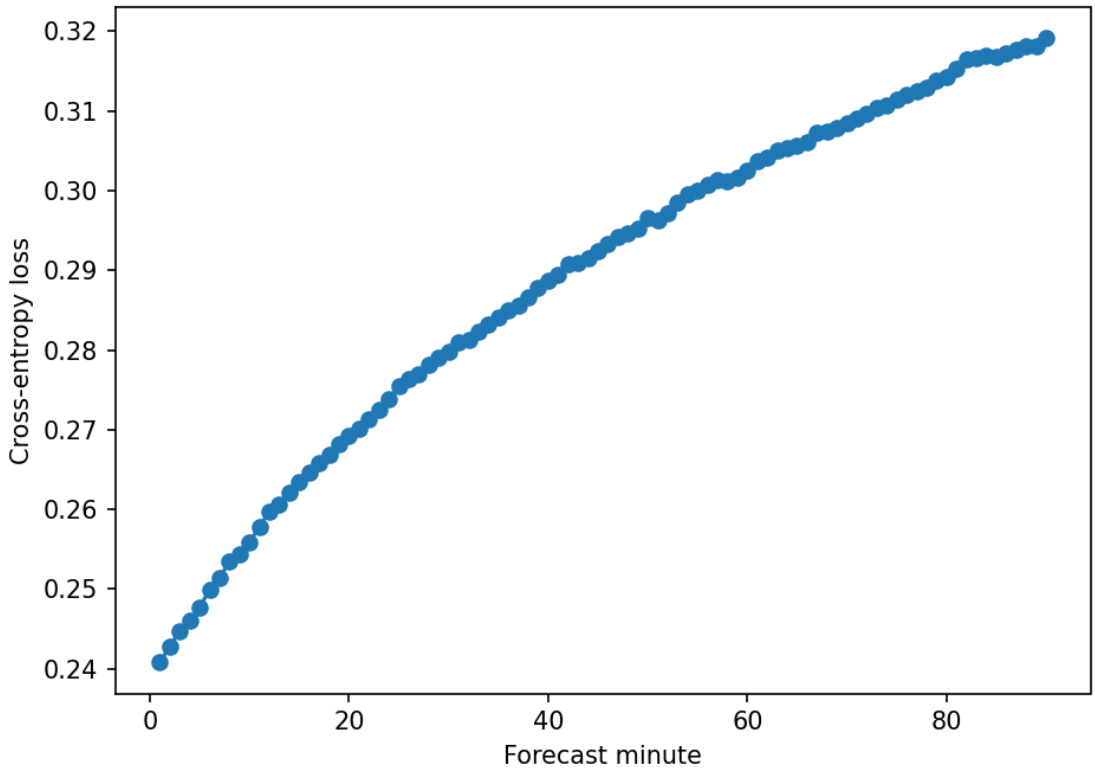

Figure 12 Cross-entropy loss along the prediction horizon.

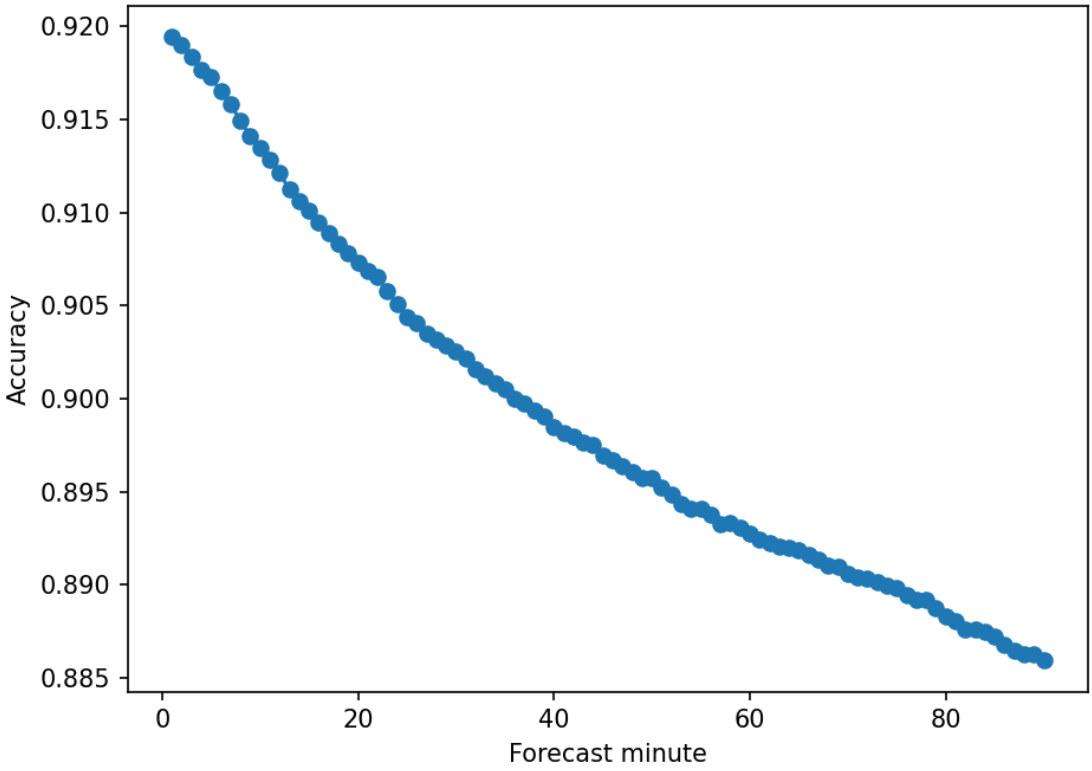

Figure 13 Accuracy along the forecast horizon.

The observed degradation in performance across the prediction horizon is expected, as uncertainty accumulates when forecasting further into the future. Nevertheless, the deterioration is relatively moderate, with accuracy remaining above 88% and cross-entropy loss remaining substantially below the baseline value defined in Section 3.2.2 even at the end of the 90-minute prediction horizon.

4.4 Maintenance Opportunity Windows

Interviews conducted with company personnel provided insight into the characteristics of different maintenance activities. These activities vary in terms of frequency, duration, resource requirements, and operational constraints. Within the scope of this study, the initial objective is to evaluate the potential of Maintenance Opportunity Windows (MOWs) to support Autonomous Maintenance (AM) activities. Two maintenance activities of particular interest were identified, each presenting different requirements and constraints.

The first activity is cleaning. This activity most closely resembles the concept of AM, as it is typically performed by the operators responsible for the machine. Cleaning does not require specialised equipment beyond the tools already available in the vicinity of the machine. Furthermore, no fixed schedule is associated with the activity; according to interviewed personnel, it is generally performed "whenever possible". Depending on the machine and the required intervention, cleaning activities typically require between 10 and 30 minutes. The activity is known to reduce adverse effects on production and is identified as a

fundamental component of both Total Productive Maintenance by Nakajima, (1988) and the 5S methodology by Hirano (1995)

The second identified activity is lubrication. Unlike cleaning, lubrication is performed at predefined intervals and generally requires specialised maintenance personnel. Furthermore, the required materials are not necessarily available near the machine and may need to be collected from dedicated storage locations beforehand. Nevertheless, lubrication is a critical maintenance activity, as it reduces equipment degradation and contributes to the prevention of unplanned interruptions. Given its typical duration, lubrication activities may also be accommodated within Maintenance Opportunity Windows.

The results indicate that the proposed LSTM model successfully predicts Maintenance Opportunity Windows capable of accommodating such activities. However, compared with the prediction metrics reported in the previous section, performance—as defined in Section 3.2.3—decreases considerably. This outcome is expected because the prediction of isolated machine states is no longer sufficient. Instead, a prediction is only considered successful if at least 30 consecutive minutes are correctly identified as belonging to a Maintenance Opportunity Window.

Using this criterion, 51.906% of the predicted Maintenance Opportunity Windows corresponded to actual Maintenance Opportunity Windows. The identified opportunities were primarily associated with the Sorting Conveyor, Inspection Conveyor, and Planer. No feasible Maintenance Opportunity Windows were identified for the Sawing Equipment.

This result is consistent with the role of the Sawing Equipment within the production system. As the primary processing bottleneck, interruptions in this asset tend to propagate rapidly throughout the system, reducing the likelihood of extended inactive periods that could be exploited for maintenance activities. In contrast, downstream assets are more likely to experience periods of starvation or blockage, creating opportunities for maintenance interventions without negatively affecting overall throughput.

5
Discussion

5.1 Overview of the results and contrast with existing literature

Previous research by Bokrantz et al. (2025) proposed an LSTM-based approach for the prediction of Maintenance Opportunity Windows (MOWs). However, the model focused on directly predicting fixed-length opportunity windows. The approach developed in this thesis differs in that it predicts the future states of the production assets, after which MOWs are identified through post-processing of the predicted machine states. A initial concern in the present research where big buffer levels were observed, is whether MOWs could be predicted in a very decoupled system, since big buffer levels could absorb the disruptions (Gershwin & Schor, 2000) that lead to the emergence of MOWs

This modification provides several advantages. First, it enables greater flexibility when identifying MOWs, since windows of different durations can be detected rather than restricting the prediction to a predefined fixed length. Consequently, maintenance activities with different duration requirements can be considered simultaneously. Second, the prediction of machine states provides additional information beyond the existence of a MOW, potentially allowing future systems to distinguish between different causes of inactivity, such as starvation, downstream blockage, or planned stops.

The adopted approach also enables a more detailed evaluation of model performance. Rather than relying solely on aggregate performance metrics, machine-specific metrics and horizon-based metrics can be computed. Machine-specific metrics make it possible to identify which production assets are more difficult to predict, while horizon-based metrics provide insight into how prediction quality evolves as the forecast extends further into the future. This information is particularly valuable when evaluating the practical applicability of the predictions, as maintenance planners may require different confidence levels depending on how far in advance a decision must be taken.

Within the scope of this thesis, the identified MOWs were evaluated with respect to two maintenance activities: cleaning and lubrication. These activities were selected because they represent common maintenance tasks with different operational requirements. Nevertheless, the proposed framework is not restricted to these activities. Any maintenance operation whose duration and resource requirements are known could potentially be evaluated against the predicted

MOWs. Consequently, the framework should be viewed as a general mechanism for identifying maintenance opportunities rather than as a tool dedicated to a specific maintenance task.

5.2 RQ1: What criteria must be met for a MOW to be considered feasible?

The results indicate that the feasibility of a Maintenance Opportunity Window (MOW) is primarily determined by time-related constraints. In its simplest form, a MOW can be considered feasible if it is sufficiently long to accommodate at least one maintenance activity from a predefined set of candidate activities. This set of activities may either be fixed or dynamically adjusted according to operational needs, such as the elapsed time since the activity was last performed.

However, maintenance activities involve more than the execution time of the intervention itself. While some activities can be carried out immediately by nearby operators using locally available tools, others require maintenance personnel to travel to the location, gather tools and materials, or interrupt other ongoing tasks. Furthermore, there is always a delay between the moment a MOW is predicted and the moment the relevant personnel become aware of it and are able to react.

Consequently, the feasibility of a MOW should not be assessed solely on the basis of the duration of the maintenance activity. Instead, sufficient time must also be available for personnel to become aware of the opportunity, prepare the required resources, and reach the equipment. This additional period can be defined as the *reaction time*.

As illustrated in Figure 14, a MOW is therefore only feasible if its duration equals at least the combined duration of the maintenance activity and the associated reaction time. This concept provides a practical bridge between predictive models and real maintenance operations, as it transforms a purely temporal prediction into an actionable maintenance opportunity.

Bokrantz et al. (2025) consider machine inactivity to be a sufficient condition for the existence of a Maintenance Opportunity Window, provided that the inactivity period meets a minimum duration requirement. This definition follows the active and inactive machine concepts introduced by Roser et al. (2001). However, not all causes of inactivity necessarily represent feasible MOWs. For example, planned stops are considered periods of inactivity, yet the personnel responsible for performing AM activities may not be present at the workstation during such periods. Consequently, inactivity resulting from certain causes should be excluded when identifying Maintenance Opportunity Windows.

The approach proposed in the present study predicts individual machine states rather than relying solely on an active/inactive classification. The

additional granularity provided by this representation offers greater flexibility, allowing users to define which machine states may constitute Maintenance Opportunity Windows and which should be disregarded.

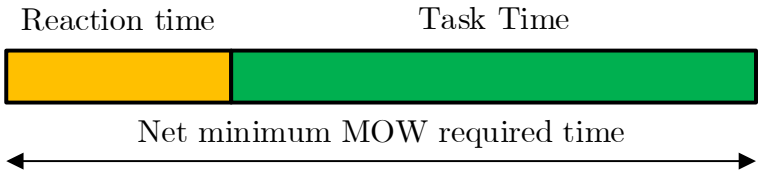

Figure 14 Illustration of the required time for MOW feasibility.

Prediction credibility constitutes a second requirement for feasibility. Even if a predicted MOW is theoretically long enough to accommodate a maintenance task, it cannot be considered operationally useful if the prediction itself is unreliable. Determining what level of model performance is sufficient for industrial implementation remains an open question and will depend on the organisation's risk tolerance and the nature of the maintenance activity. Nevertheless, the methodology proposed in this thesis demonstrates that specialised performance metrics can support this assessment. In particular, machine-specific metrics and horizon-based metrics provide considerably more insight than aggregate averages, allowing practitioners to evaluate prediction reliability under different operational conditions.

Ultimately, MOW feasibility should be understood as the combination of two dimensions: temporal feasibility and predictive credibility. Both conditions must be satisfied before maintenance activities can be confidently allocated to a predicted opportunity window.

5.3 RQ2: How can MOWs be predicted in a sawmill industry context?

The results demonstrate that LSTM models constitute a viable approach for predicting Maintenance Opportunity Windows in a sawmill production environment. Rather than directly predicting MOWs, the proposed methodology predicts the future states of the production assets and subsequently identifies MOWs through post-processing. This approach proved capable of generating forecasts that remain useful throughout a 90-minute prediction horizon.

The study also provides insight into the data requirements necessary to achieve this performance. The results suggest that complex condition-monitoring systems are not strictly necessary for an initial implementation. Instead, the most important inputs are the operational states of the production assets and variables representing the time elapsed since the occurrence of specific events, such as random stops, setups, or planned stops. Since these variables can be derived from relatively simple production data, the required data-collection infrastructure is substantially less demanding than many predictive maintenance applications

based on vibration analysis, thermal monitoring, or other advanced sensing technologies. This characteristic may facilitate adoption in organisations with lower maintenance maturity levels, according to the description by by Oliveira & Lopes (2019), where advanced condition-monitoring infrastructure may not yet be available. Consequently, MOW prediction may represent an accessible entry point towards more data-driven maintenance practices.

An important methodological finding concerns the evaluation procedure itself. Initial experiments based on a conventional train-validation split produced highly encouraging results. However, validation using completely independent simulation logs revealed substantially poorer performance, indicating a strong tendency towards overfitting. This suggests that conventional validation approaches may overestimate model performance when simulation-generated data is used. Consequently, independent simulation runs should be preferred when validating similar prediction models.

The selection of sequence length and prediction horizon also appears to be strongly linked to the characteristics of the production system and the intended maintenance activities. The sequence length should be sufficiently long to capture the relevant dependencies between upstream and downstream assets. In the present case, a sequence length of 10,000 minutes was found to provide the model with sufficient contextual information to capture interactions spanning the entire production system. Subramaniyan et al (2018a) suggest that the quality of the prediction might be sensitive to the amount of past data used; this supports the idea that the chosen amount of past data to use at each run is based on cycle times and not chosen arbitrarily.

Similarly, the prediction horizon should not be determined solely according to model accuracy. Instead, it should reflect the practical requirements of the maintenance activities that are expected to exploit the predicted MOWs. Activities such as cleaning may only require a short advance notice, whereas interventions requiring specialised personnel, spare parts, or preparation activities may benefit from significantly longer horizons. Consequently, an optimal prediction horizon represents a trade-off between predictive accuracy and operational usefulness.

From an implementation perspective, the results are encouraging. Once trained, the model is capable of generating a 90-minute forecast in less than one second using consumer-grade hardware (Nvidia RTX3060, 6GB). This suggests that real-time implementation would not require particularly expensive computational infrastructure. In practice, the computational effort associated with data collection, preprocessing, and communication may exceed the prediction time itself. Nevertheless, the results indicate that frequent forecast updates could be performed without imposing a significant computational burden.

5.4 RQ3: How can simulation be used to define the requirements of the proposed prediction model?

One of the principal challenges associated with machine-learning applications in industrial environments is the availability of suitable data. Data collection systems are often limited, incomplete, or entirely absent when a project begins. Furthermore, obtaining sufficient data may require substantial investments in sensors, communication infrastructure, and storage systems, followed by lengthy periods of operation before enough information has been accumulated for model development.

The results of this study demonstrate how simulation can mitigate these challenges. By generating synthetic production data, the simulation model enabled the development and evaluation of the prediction model without requiring any initial investment in physical data-collection infrastructure. In the present case, approximately 720 days of production data were generated in around six hours of simulation execution, illustrating the substantial acceleration that simulation can provide during the development phase. This contrasts with the work of Bokrantz et al. (2025), where two years of production data were already available for model development. In the present case, the absence of historical production data could initially have been perceived as a significant limitation. However, the simulation model enabled this limitation to be overcome by generating an equivalent amount of data within a relatively short period of time.

Beyond simply generating data, simulation also proved valuable for identifying data requirements. Because the exact information required by the prediction model was initially unknown, the simulation environment allowed variables to be added, removed, and modified iteratively. This process made it possible to determine which inputs contributed most to prediction performance before defining the requirements of a real-world data-collection system. Consequently, simulation acted not only as a source of synthetic data but also as a tool for reducing uncertainty regarding future investments.

The use of simulation also highlights a broader opportunity. While the present study focuses on predicting Maintenance Opportunity Windows, simulation environments could support the development of more advanced decision-support systems. In particular, simulation could provide a safe environment for testing maintenance-planning strategies and evaluating their impact on production performance before deployment in real production systems. Although such applications fall outside the scope of this thesis, the results suggest that simulation can play a central role in bridging the gap between machine-learning research and industrial implementation.

5.5 Methodological Limitations

The proposed methodology presents several limitations that should be acknowledged. Subramaniyan et al (2018a) warn that models trained and validated exclusively using simulation data may initially exhibit overly optimistic performance that is difficult to replicate in real industrial environments. This raises concerns regarding the direct implementation of the proposed model, as its performance may decrease when exposed to real production data unless additional training or fine-tuning is performed using industrial datasets.

This observation also highlights a potential methodological dilemma. Simulation can be used to overcome situations in which production data is unavailable, insufficient, or where the data requirements for a machine-learning application are not yet known. In this study, simulation demonstrated being a tool able to iteratively define data requirements, and enabled the generation of the large datasets required to develop and evaluate the prediction model. However, the final industrial implementation of the approach would still require the collection of the necessary production data, implying that simulation can mitigate the data-availability problem during development but cannot completely eliminate the need for real-world data in the long term.

Another limitation concerns the representation of the drying process. In the simulation model, continuous dryers were modelled as static dryers. Although this approach ensures similar throughput levels over sufficiently long periods of time, important differences emerge when analysing shorter time horizons. In particular, the material flow feeding the inspection conveyor differs significantly between the two representations. Continuous dryers behave more similarly to conveyors, releasing smaller batches of material at relatively short and regular intervals. In contrast, static dryers remain idle until their maximum capacity has been reached, after which processing begins, and subsequently release substantially larger batches at a much lower frequency, increasing the inter-arrival time between batches. While both approaches may provide similar overall throughput in the long term, the short-term dynamics of material flow differ considerably. Consequently, the frequency and duration of starvation periods in downstream equipment may be affected, potentially influencing the occurrence and length of predicted Maintenance Opportunity Windows.

Several studies define bottleneck resources based on the concept of machine activity, where bottlenecks are identified as the machines exhibiting the longest active periods within the production system (Roser et al., 2001, 2002; Subramaniyan et al., 2016). Subramaniyan et al. (2020) explicitly relate this behaviour to the fact that non-bottleneck resources are more susceptible to starvation and blockage. Since Maintenance Opportunity Windows are inherently associated with machine inactivity, MOWs are more likely to emerge in non-bottleneck resources than in bottlenecks.

This observation raises an interesting discussion regarding the potential impact of Opportunistic Maintenance on overall system performance. While exploiting MOWs may improve the condition and availability of non-bottleneck resources at a limited production cost, potentially reducing their probability of becoming a shifting bottleneck, systems characterised by predominantly static bottlenecks might find larger improvements in throughput through approaches that focus more directly on bottleneck resources. The evaluation of such alternatives falls outside the scope of the present study but represents an interesting topic for future research.

5.6 Recommendations for Data Collection Systems

One of the objectives of the proposed methodology was to identify the data requirements necessary for future industrial implementation. While simulation was used to generate synthetic data during this study, a production deployment would require the collection of equivalent real-world data.

The most important requirement is the continuous logging of machine states. Each monitored asset should be capable of recording transitions between production, planned stop, random stop, setup, starvation, and downstream blockage states, together with timestamps. Later on, the collected data can be processed and transformed into time-series data, more suitable for ML applications.

Random stops should be classified according to their underlying causes rather than being recorded as a single generic state. This would allow the development of a more precise model in the future, since different causes trigger with different frequencies and incur into different downtimes.

Importantly, the results of this study suggest that the implementation of a MOW prediction system does not require an extensive condition-monitoring infrastructure. The proposed approach relies primarily on production-state information that can be obtained with relatively limited investments in data collection.

5.7 Human-Centric Considerations

The successful implementation of a MOW prediction system would require an effective mechanism for communicating predictions to operators and maintenance personnel. The value of a predicted Maintenance Opportunity Window depends not only on the quality of the prediction itself, but also on the ability of personnel to interpret the information and act upon it. Consequently, suitable visualisation and notification systems should be regarded as essential components of any future industrial implementation.

Ahmed Murtaza et al. (2024) highlight the importance of Human-Machine Interfaces (HMIs) within the context of Industry 5.0. According to the authors,

effective HMIs should provide clear and intuitive outputs while also allowing users to contribute information and feedback to the system. In the context of MOW prediction, this suggests that visualisation systems should prioritise simplicity and usability. Although detailed reports regarding prediction performance may be valuable for model developers and maintenance engineers, production personnel are likely to benefit more from concise information that can be rapidly interpreted within the fast-paced environment of a production line. For example, upcoming Maintenance Opportunity Windows could be displayed together with their expected duration and estimated starting time, avoiding unnecessary complexity in the presentation of the information.

Industry 5.0 also emphasises collaboration between humans and intelligent systems rather than complete automation. In our case, this can be translated as operators and maintenance personnel being able to provide feedback regarding the quality of the predictions. For example, users could indicate whether a predicted MOW occurred as expected or whether an unpredicted MOW emerged. Such feedback could be collected through simple interactions with the HMI. One possible implementation could consist of a single button conveniently located below the display, allowing operators to register the occurrence of a Maintenance Opportunity Window with minimal disruption to their normal activities. The collected information could subsequently be used to evaluate model performance, identify concept drift, and determine when retraining or refinement of the prediction model may be necessary.

Human oversight should remain an integral part of the decision-making process despite the automation opportunities enabled by data-driven maintenance approaches. (Ahmed Murtaza et al., 2024). A future implementation could integrate the MOW prediction system with Enterprise Resource Planning (ERP) and Computerised Maintenance Management System (CMMS) platforms to suggest maintenance activities suitable for upcoming windows, attending optimality on criteria such as resource management, life-cycle extension and downtime minimisation, hence improving sustainability of the company. However, the final decision regarding which activity should be performed should remain within maintenance personnel, who are better positioned to consider the operational context, resource availability, and practical constraints that may not be fully captured by the prediction system.

5.8 Impact of the Research

5.8.1 Impact on the Sawmill Industry

The results of this study suggest that MOW prediction may constitute a viable approach for improving maintenance planning within the sawmill industry. As discussed previously, advanced maintenance approaches remain relatively uncommon in the sector, where maintenance activities are often based on reactive

and preventive strategies. The proposed methodology demonstrates that useful maintenance predictions can be generated using relatively simple production-state data, reducing the need for extensive condition-monitoring infrastructure. Furthermore, by enabling maintenance activities to be performed during naturally occurring production interruptions, the approach may contribute to increasing equipment availability and reducing lost sold value, which Thomas & Weiss (2021) identify as one of the principal consequences of machine unavailability.

5.8.2 Contributions to Maintenance Research

From a maintenance perspective, this research contributes to the emerging Flow View of Opportunistic Maintenance proposed by Bokrantz et al. (2025). While existing research has primarily focused on how Maintenance Opportunity Windows can be exploited once identified, the present study addresses the problem of predicting such opportunities before they occur. On a broader spectrum, the ability to anticipate future maintenance opportunities supports the alignment of maintenance activities with production requirements, which is consistent with the maintenance-management principles discussed by Salonen & Bengtsson (2011). Furthermore, the proposed approach aligns with the broader transition from reactive towards proactive maintenance strategies described by Pintelon et al. (2006), as maintenance activities can be planned in anticipation of future production conditions rather than in response to disruptions after they occur.

5.8.3 Sustainability Implications

The proposed methodology may also contribute to sustainability objectives. By increasing the ability to perform maintenance without disrupting production, the approach can support higher equipment availability and improved utilisation of existing production assets. Improved maintenance planning may also reduce unnecessary interventions, extend equipment lifetime, and lower the resources required to recover from failures. Although the sustainability benefits were not quantified within this study, the reduction of production losses and the more efficient use of industrial resources are consistent with broader sustainability objectives within manufacturing systems.

5.9 Future work

The present study demonstrates the feasibility of predicting Maintenance Opportunity Windows (MOWs) in a sawmill production context. Nevertheless, several opportunities exist for extending both the simulation framework and the prediction methodology developed throughout this thesis.

Concerning the simulation framework, one natural extension would be the inclusion of transportation activities within the production system. The movement of materials, particularly in industries such as sawmilling where

products are large and heavy, often requires specialised transportation equipment. Such equipment constitutes a shared resource whose availability may influence production performance. Depending on the operating conditions, transportation resources may temporarily become bottlenecks, constraining throughput despite sufficient processing capacity elsewhere in the system. Incorporating transportation into the simulation model would therefore increase its realism and improve its ability to capture interactions between production assets and material-handling resources.

Furthermore, the simulation framework could be expanded with standardised methods for measuring and analysing buffer occupancy, machine utilisation, and other performance indicators over time. Such capabilities would facilitate the use of the framework for bottleneck detection and production-system analysis beyond the scope of MOW prediction.

Another area for improvement concerns the representation of random stops. In the current implementation, all random stop causes are grouped under a single status label within the production logs. As a consequence, different underlying phenomena characterised by distinct statistical behaviours are represented identically from the perspective of the machine-learning model. This limits the amount of information available to the predictor. Future developments could distinguish between different categories of random stops and maintain separate time-since-occurrence variables for each category. Such an approach would likely improve the model's ability to identify recurring patterns and increase prediction accuracy.

Several interviewees also suggested that the production schedule governing both log-pile selection and sawing-pattern allocation may significantly influence the bottleneck behaviour of the production system. This observation merits further investigation. In particular, optimisation approaches could be explored to identify scheduling policies that improve throughput, reduce work-in-progress inventory, or create additional maintenance opportunities without requiring physical modifications to the production system.

Concerning the prediction model itself, future developments could extend the framework beyond the passive prediction of MOWs. Rather than merely identifying naturally occurring maintenance opportunities, a decision-support system could actively generate them by temporarily halting selected production assets when buffer levels indicate that upstream and downstream sections of the production system can continue operating without affecting overall throughput. Such an approach would transform the model from a predictive tool into a prescriptive one, enabling it to recommend actions that improve maintenance planning and operational performance. The prediction model could be additionally integrated with ERP and CMMS systems, allowing optimal scheduling of maintenance operations ensuring efficient use of resources and focusing on asset life extension and downtime minimisation.

Additionally, the robustness of the prediction model could be further evaluated under a wider range of operating conditions. The current work focuses primarily on the baseline production scenario, whereas future studies could assess model performance under the alternative scenarios explored during the Monte Carlo analysis, such as different buffer configurations, material shortages, or modified production schedules. This would provide greater confidence regarding the generalisability of the proposed approach.

Finally, future work could explore a closer integration between MOW prediction and CBM or PdM strategies. Crespo del Castillo & Parlikad (2024), propose scheduling maintenance activities within existing maintenance opportunities as close as possible to their predicted due dates without exceeding them. Integrating such concepts with the MOW prediction framework developed in this thesis would enable maintenance activities to be scheduled according to both asset condition and production constraints. This could improve maintenance efficiency by reducing unnecessary interventions while simultaneously minimising production disruptions. Ultimately, such integration would move maintenance planning from a reactive or calendar-based approach towards a more adaptive and data-driven decision-making process. The simulation model and the MOWs prediction model could be integrated into a comprehensive Digital Shadow or Digital Twin of the production system, continuously updated with real-time data, enabling dynamic evaluation of MOWs and real-time optimisation of maintenance schedules.

6
Conclusion

This thesis investigated the feasibility of enabling Opportunistic Maintenance in a sawmill environment through the prediction of Maintenance Opportunity Windows (MOWs). To achieve this, a discrete-event simulation model of a generic sawmill was developed to generate production data, which was subsequently used to train a Long Short-Term Memory (LSTM) model capable of predicting future machine states and identifying potential maintenance opportunities.

The results demonstrate that Maintenance Opportunity Windows can be predicted using simple production-state data. The developed LSTM model achieved consistently strong predictive performance throughout the evaluated prediction horizon, showing that future machine states can be forecasted with sufficient accuracy to support maintenance planning. By predicting machine states rather than predefined maintenance windows, the proposed approach provides greater flexibility in identifying opportunities of varying duration and operational relevance.

Regarding the first research question, the study found the feasibility requirements for a MOW. A feasible MOW must satisfy a minimum duration requirement while also considering the operational context of the inactive state. Consequently, the practical applicability of a maintenance opportunity depends not only on its length but also on whether the inactivity originates from conditions that genuinely allow maintenance work to be performed.

Regarding the second research question, the results show what data is needed to predict MOWs in a sawmill context. The proposed methodology demonstrates the potential of data-driven approaches for supporting opportunistic maintenance in this environment.

Regarding the third research question, the study demonstrates that simulation can be used effectively to define the data requirements of predictive maintenance solutions. The simulation model enabled the generation of large production datasets, supported the identification of relevant input variables, and provided a practical means of developing and evaluating the prediction model prior to investments in industrial data-collection infrastructure.

Overall, the thesis demonstrates the potential of combining simulation and machine learning to support Opportunistic Maintenance in the sawmill industry. The proposed framework provides both a foundation for future industrial implementation and a basis for further research on Maintenance Opportunity

Window prediction and maintenance planning in data-limited production environments.

References

Ab-Samat, H., & Kamaruddin, S. (2014). Opportunistic maintenance (OM) as a new advancement in maintenance approaches: A review. *Journal of Quality in Maintenance Engineering*, *20*(2), 98–121. https://doi.org/10.1108/JQME-04-2013-0018

Ahmed Murtaza, A., Saher, A., Hamza Zafar, M., Kumayl Raza Moosavi, S., Faisal Aftab, M., & Sanfilippo, F. (2024). Paradigm shift for predictive maintenance and condition monitoring from Industry 4.0 to Industry 5.0: A systematic review, challenges and case study. *Results in Engineering*, *24*, 102935. https://doi.org/10.1016/j.rineng.2024.102935

Banks, J., Carson, J. S., Nelson, B., & Nicol, D. M. (2010). *Discrete event system simulation* (5th edn). Pearson.

Bishop, C. M. (2006). Neural Networks—Model Training. In *Pattern Recognition and Machine Learning* (1st edn, pp. 252–261). Springer. https://link.springer.com/book/9780387310732

Bokrantz, J., Subramaniyan, M., & Skoogh, A. (2025). Seizing opportunity: Advancing the science and practice of opportunistic maintenance in manufacturing. *International Journal of Production Economics*, *288*, 109704. https://doi.org/10.1016/j.ijpe.2025.109704

Burnes, B. (1996). No such thing as ... a "one best way" to manage organizational change. *Management Decision*, *34*(10), 11–18. https://doi.org/10.1108/00251749610150649

Colledani, M., Magnanini, M. C., & Tolio, T. (2018). Impact of opportunistic maintenance on manufacturing system performance. *CIRP Annals*, *67*(1), 499–502. https://doi.org/10.1016/j.cirp.2018.04.078

Crespo del Castillo, A., & Parlikad, A. K. (2024). Dynamic fleet management: Integrating predictive and preventive maintenance with operation workload balance to minimise cost. *Reliability Engineering & System Safety*, *249*, 110243. https://doi.org/10.1016/j.ress.2024.110243

Eliasson, E. (2026, May). *Structural Tightening in Europe's Timber Market – Why Supply Is No Longer Following Demand*. Norra Timber. https://www.norratimber.com/news/wood-market/structural-tightening-in-europe-s-timber-market-why-supply-is-no-longer-following-demand/

Foumani, N. M., Miller, L., Tan, C. W., Webb, G. I., Forestier, G., & Salehi, M. (2024). Deep Learning for Time Series Classification and Extrinsic

Regression: A Current Survey. *ACM Comput. Surv.*, *56*(9), 217:1-217:45. https://doi.org/10.1145/3649448

From raw material to wood product. (2025). Swedish Wood. https://www.swedishwood.com/wood-facts/about-wood/from-log-to-plank/

Garg, A., & Deshmukh, S. G. (2006). Maintenance management: Literature review and directions. *Journal of Quality in Maintenance Engineering*, *12*(3), 205–238. https://doi.org/10.1108/13552510610685075

Gershwin, S. B., & Schor, J. E. (2000). Efficient algorithms for buffer space allocation. *Annals of Operations Research*, *93*(1), 117–144. https://doi.org/10.1023/A:1018988226612

Giacotto, A., Marques, H. C., & Martinetti, A. (2025). Prescriptive maintenance: A comprehensive review of current research and future directions. *Journal of Quality in Maintenance Engineering*, *31*(1), 129–173. https://doi.org/10.1108/JQME-07-2023-0064

Hirano, H. (1995). *5 Pillars of the Visual Workplace*. Productivity Press. https://doi.org/10.4324/9780367804886

Iung, B., Do, P., Levrat, E., & Voisin, A. (2016). Opportunistic maintenance based on multi-dependent components of manufacturing system. *CIRP Annals*, *65*(1), 401–404. https://doi.org/10.1016/j.cirp.2016.04.063

Kolambe, M., & Arora, S. (2024). Forecasting the Future: A Comprehensive Review of Time Series Prediction Techniques. *Journal of Electrical Systems*, *20*(2s), 575–586. https://doi.org/10.52783/jes.1478

Krauß, J., Hülsmann, T., Leyendecker, L., & Schmitt, R. H. (2023). Application Areas, Use Cases, and Data Sets for Machine Learning and Artificial Intelligence in Production. *Production at the Leading Edge of Technology*, 504–513. https://doi.org/10.1007/978-3-031-18318-8_51

Li, W., & Li, T. (2025). Comparison of deep learning models for predictive maintenance in industrial manufacturing systems using sensor data. *Scientific Reports*, *15*(1), 23545. https://doi.org/10.1038/s41598-025-08515-z

Linnaeus University. (2025). *Project description for Simulation Course*.

Naidu, G., Zuva, T., & Sibanda, E. M. (2023). A Review of Evaluation Metrics in Machine Learning Algorithms. In R. Silhavy & P. Silhavy (Eds), *Artificial Intelligence Application in Networks and Systems* (pp. 15–25). Springer International Publishing. https://doi.org/10.1007/978-3-031-35314-7_2

Nakajima, S. (1988). *Introduction to TPM: Total Productive Maintenance*. Productivity Press. https://es.scribd.com/document/535428942/Introduction-to-TPM-Total-Productive-Maintenance

Nicolai, R. P., & Dekker, R. (2008). Optimal Maintenance of Multi-component Systems: A Review. In K. A. H. Kobbacy & D. N. P. Murthy (Eds), *Complex System Maintenance Handbook* (pp. 263–286). Springer. https://doi.org/10.1007/978-1-84800-011-7_11

Oliveira, M. A., & Lopes, I. (2019). Evaluation and improvement of maintenance management performance using a maturity model. *International Journal of Productivity and Performance Management*, *69*(3), 559–581. https://doi.org/10.1108/IJPPM-07-2018-0247

Pintelon, L., & Parodi-Herz, A. (2008). Maintenance: An Evolutionary Perspective. In K. A. H. Kobbacy & D. N. P. Murthy (Eds), *Complex System Maintenance Handbook* (pp. 21–48). Springer. https://doi.org/10.1007/978-1-84800-011-7_2

Pintelon, L., Pinjala, S. K., & Vereecke, A. (2006). Evaluating the effectiveness of maintenance strategies. *Journal of Quality in Maintenance Engineering*, *12*(1), 7–20. https://doi.org/10.1108/13552510610654501

Roser, C., Nakano, M., & Tanaka, M. (2001). A practical bottleneck detection method. *Proceeding of the 2001 Winter Simulation Conference (Cat. No.01CH37304)*, *2*, 949–953 vol.2. https://doi.org/10.1109/WSC.2001.977398

Roser, C., Nakano, M., & Tanaka, M. (2002). Shifting bottleneck detection. *Proceedings of the Winter Simulation Conference*, *2*, 1079–1086 vol.2. https://doi.org/10.1109/WSC.2002.1166360

Salonen, A., & Bengtsson, M. (2011). The potential in strategic maintenance development. *Journal of Quality in Maintenance Engineering*, *17*(4), 337–350. https://doi.org/10.1108/13552511111180168

Södra skogsägarna. (n.d.). *Södra—About us*. Retrieved 5 May 2026, from https://www.sodra.com/en/uk-ie/about-us/

Subramaniyan, M., Skoogh, A., Gopalakrishnan, M., Salomonsson, H., Hanna, A., & Lämkull, D. (2016). An algorithm for data-driven shifting bottleneck detection. *Cogent Engineering*, *3*(1), 1239516. https://doi.org/10.1080/23311916.2016.1239516

Subramaniyan, M., Skoogh, A., Muhammad, A. S., Bokrantz, J., Johansson, B., & Roser, C. (2020). A data-driven approach to diagnosing throughput

bottlenecks from a maintenance perspective. *Computers & Industrial Engineering*, *150*, 106851. https://doi.org/10.1016/j.cie.2020.106851

Subramaniyan, M., Skoogh, A., Salomonsson, H., Bangalore, P., & Bokrantz, J. (2018). A data-driven algorithm to predict throughput bottlenecks in a production system based on active periods of the machines. *Computers & Industrial Engineering*, *125*, 533–544. https://doi.org/10.1016/j.cie.2018.04.024

Subramaniyan, M., Skoogh, A., Salomonsson, H., Bangalore, P., Gopalakrishnan, M., & Sheikh Muhammad, A. (2018). Data-driven algorithm for throughput bottleneck analysis of production systems. *Production & Manufacturing Research*, *6*(1), 225–246. https://doi.org/10.1080/21693277.2018.1496491

Swedish Forest Industries Federation. (2025, October 30). *Industry Calls for Regulatory Clarity and Reduced Burden in EU Deforestation Regulation—Swedish Forest Industries Federation*. https://www.forestindustries.se/news/latest-news/2025/10/industry-calls-for-regulatory-clarity-and-reduced-burden-in-eu-deforestation-regulation/

Thomas, D. S., & Weiss, B. (2021). Maintenance Costs and Advanced Maintenance Techniques: Survey and Analysis. *International Journal of Prognostics and Health Management*, *12*(1). https://doi.org/10.36001/ijphm.2021.v12i1.2883

van Rossum, G., Warsaw, B., & Coghlan, A. (2013). *PEP 8 – Style Guide for Python Code / Naming Conventions*. Python Enhancement Proposals (PEPs). https://peps.python.org/pep-0008/#naming-conventions

Vu, H. C., Do, P., Fouladirad, M., & Grall, A. (2020). Dynamic opportunistic maintenance planning for multi-component redundant systems with various types of opportunities. *Reliability Engineering & System Safety*, *198*, 106854. https://doi.org/10.1016/j.ress.2020.106854

Wen, X., & Li, W. (2023). Time Series Prediction Based on LSTM-Attention-LSTM Model. *IEEE Access*, *11*, 48322–48331. https://doi.org/10.1109/ACCESS.2023.3276628

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). 10. Modern Recurrent Neural Networks. In *Dive into Deep Learning*. Cambridge University Press. https://d2l.ai/chapter_recurrent-modern/lstm.html

Appendix A: Description of generic simulation assets

Table 4 Essential attributes of MeasuredEnvironment.

Name	Category	Description
time_unit	R	Defines the base time unit used by the simulation environment. Must be “seconds”, “minutes” or “hours”
_unit_seconds	C	Stores the conversion factor between the selected time_unit and seconds.
_precision_digits	R	Defines the number of decimal places used when applying numerical precision corrections.

Table 5 Essential methods of MeasuredEnvironment.

Name	Description
_round_for_env	Utility method used to apply consistent rounding according to _precision_digits.
convert_planned_stops	Converts planned-stop intervals expressed in seconds, minutes, or hours into the internal simulation time unit.
convert_planned_stop_loop_length	Converts the planned-stop cycle length from seconds, minutes, or hours into the internal simulation time unit.
schedule	Overrides the standard event-scheduling mechanism to improve numerical precision when advancing simulation time.

Table 6 Essential attributes and inputs of SimpyRNG.

Name	Category	Description
special_seed	O	Optional seed used to initialise the random-number generator.
rng	O+C	If None, a new random-number generator is automatically created using special_seed when

	provided. Alternatively, an existing generator object may be supplied directly. The recommended approach is to provide special_seed and leave rng as None.

Table 7 Essential methods of RNG.

Name	Description
gen_random	Abstract placeholder method intended to be overridden by subclasses. Different parameters may be accepted depending on the implementation (e.g., generating multiple values in a single call).

Table 8 Bundled variants of SimpyRNG.

Name	Description
Constant	Allows constant values to be used wherever a random-number generator is expected. Randomness has no effect.
RandomWeibull	Weibull distribution supporting lower and upper bounds.
RandomExponential	Exponential distribution implemented as a Weibull distribution with shape = 1, bounded within $[0, +\infty)$.
RandomUniform	Uniform distribution supporting lower and upper bounds.
Normal	Normal distribution supporting lower and upper bounds.
Iterator	Cyclically returns elements from an iterable. Randomness has no effect.
ConstantAttributeBased	Given an attribute name and a rule set, extracts the specified attribute from one or more objects and returns the corresponding value according to the defined rules. Randomness has no effect.

Table 9 Essential attributes of Material.

Name	Category	Description
type_id	R	Identifier associated with the created Material object. The value may take any form deemed appropriate by the modeller (e.g., "Plank Type X").

volume	C	Defaults to 1. The value is calculated through volume_calculation() and may be overridden in subclasses where volume is relevant.
_debug_id	C	Automatically assigned unique identifier used for debugging and traceability purposes.

Table 10 Essential methods of Material.

Name	Description
volume_calculation	Method responsible for determining the volume of the Material. The signature accepts any *args provided during initialisation and is intended to be overridden in subclasses where volume calculations are required.

Table 11 Essential methods of GenericLogger.

Name	Description
write_log	Method responsible for recording information whenever a logging event occurs. Production-asset classes invoke this method to store relevant simulation data.
finalise	Method intended to be executed at the end of a simulation run. Responsible for processing the collected data and transforming it into the desired exportable format.

Table 12 Essential attributes of GenericElement.

Name	Category	Description
env	R	Reference to the simulation environment. Must be an instance of MeasuredEnvironment. This attribute enables interaction with the simulation clock and allows the asset to create and schedule events.
capacity	R	Maximum number of Material objects that may coexist within the asset simultaneously. Internally, this parameter is used to initialise the storage structure responsible for managing incoming Material objects (but not the actual use of capacity).

logger	O	Optional instance of GenericLogger used to generate simulation logs.
process_time_distribution	O	Optional SimpyRNG used to generate processing times for incoming Material objects.
failure_distribution	O	Optional dictionary with type hint dict[str, SimpyRNG]. Keys represent random-stop causes, while values define the distributions used to generate the time until the next occurrence of each cause.
repair_distribution	O	Follows the same structure as failure_distribution, but defines the duration of each random stop rather than its occurrence time.
planned_stops	O+C	Defines recurring planned-stop intervals. When provided, the intervals are converted automatically to the time unit used by the associated MeasuredEnvironment. This attribute is closely related to planned_stop_loop_length. *
planned_stop_loop_length	O+C	Defines the duration of the cycle over which planned_stops repeat. When provided, the value is converted automatically to the time unit used by the associated MeasuredEnvironment. **
loads_when_halted	R	Boolean flag indicating whether new Material objects may enter the asset while it is undergoing a planned stop or random stop.
logger_id	O+C	Identifier used when interacting with the logger. May also assist debugging. If None is provided, a unique identifier is generated automatically.

_last_log_time	C	Stores the most recent simulation time at which a logging event occurred. Used internally by the logging system.
_last_log_summon_reason	C	Stores the reason that triggered the most recent logging event. Used internally to ensure correct logging behaviour.
slots	C	Internal storage structure used to manage the number of Material objects currently contained within the asset.
_planned_intervals	C	Cached representation of the intervals defined in planned_stops, created to improve performance during simulation execution.
_planned_starts	C	Cached collection containing the start time of each planned-stop interval. Used to improve lookup performance.
next_planned_stop	C	Stores the simulation time at which the next planned stop will begin.
next_failure	C	Dictionary storing the required processing time for the next occurrence of each potential random-stop cause.
next_failure_name	C	Stores the name of the random-stop cause associated with the earliest upcoming random-stop event.
next_failure_duration	C	Stores the duration of the next random-stop event.
_blocked	C	Internal flag used to temporarily prevent Material objects from entering processing routines while preserving reserved capacity. Primarily used to avoid race conditions.
_block_event	C	Event associated with _blocked, used to suspend and resume processes when required.

failed	C	Boolean flag indicating whether the asset is currently in a random-stop state.
failed_event	C	Event associated with an active random stop. The event succeeds when the random-stop period ends.
net_time_since_last_repair	C	Stores the accumulated production time since the end of the most recent random stop. Time spent in non-producing states, such as planned stops or downstream blockages, is excluded.
_coming_items	C	Internal structure used to avoid race conditions when multiple incoming Material objects are scheduled to be stored and processed simultaneously in time.
successors	C	List containing the potential downstream GenericElement instances to which Material objects may be transferred.

*The attribute planned_stops is initially provided with the type hint:
tuple[dict[tuple[float, float], bool], Literal["hours"] | Literal["minutes"] | Literal["seconds"]] | None
The expected structure is:
({(start_time, end_time): is_planned_stop}, unit).
where start_time and end_time define the boundaries of a planned-stop interval, is_planned_stop=True indicates that the interval corresponds to a planned stop, and unit specifies whether the interval is expressed in hours, minutes, or seconds.
The final interval should terminate at the same point as the cycle defined by planned_stop_loop_length, ensuring that the planned-stop schedule can repeat indefinitely.

**The attribute planned_stop_loop_length is initially provided with the type hint:
tuple[int | float, Literal["hours"] | Literal["minutes"] | Literal["seconds"]] | None
The expected structure is:
(length, unit)
where length defines the duration of the complete planned-stop cycle and unit specifies whether the value is expressed in hours, minutes, or seconds.

Table 13 Essential methods of GenericElement.

Name	Description
store	Introduces a Material object into the asset. Depending on the value of loads_when_halted, this method is assigned a different internal implementation during class initialisation (_storeitem_loads_when_halted or _storeitem_restricted_loading).
_storeitem_loads_when_halted	Internal implementation of store that allows Material objects to enter the asset while it is undergoing a planned stop or random stop.
_storeitem_restricted_loading	Internal implementation of store that prevents Material objects from entering the asset while it is undergoing a planned stop or random stop.
send_away	Transfers a Material object to another GenericElement. The method includes wrapper mechanisms that allow inherited classes to customise routing behaviour. By default, materials are transferred to the first successor in the successors list.
set_target	Wrapper method used by send_away to determine the destination of a Material object. Intended to be overridden when routing rules are required.
_write_log_unchecked	Internal method used to interact directly with the GenericLogger stored in logger. Users are not expected to call this method directly.
_planned_stop_interruptions_between	Auxiliary method used to ensure correct logging of starvation and

	downstream blockage periods when planned stops occur during those states.
_insert_mid_planned_stop_if_needed	Determines whether planned-stop interruptions must be inserted into the logging sequence, depending on the nature of the state being logged.
logger_interaction	Handles interaction with the GenericLogger. This method may be invoked at different points by inherited classes depending on their specific behaviour.
process	Placeholder method that defines how the asset interacts with incoming Material objects. Since GenericElement is not intended to be instantiated directly, this method must be implemented by subclasses. Implementations typically define processing behaviour, planned and random stop comprobations, and interaction with the simulation environment.
check_yielding_constraints	Implements the standard logic used to compare the remaining processing time with upcoming stop events, determining the next time-advancement event for the asset.
calculate_yield_time	Computes the remaining processing time before the current operation can continue.
generate_process_time	Generates the processing time associated with a newly received Material object.
generate_next_failure	Determines the cause and occurrence time of the next random stop.

fail_block	Handles the activation of a random stop by creating the relevant event and updating the associated state variables.
fail_unblock	Handles the termination of a random stop by resolving the relevant event and restoring the corresponding state variables.
fail	Executes the complete random-stop mechanism. This includes triggering the stop, scheduling its duration, restoring operation afterwards, and generating the next random-stop event.
block_store_before_evaluation	Activates _blocked and _block_event to temporarily prevent incoming Material objects from entering processing routines while evaluations are being performed.
unblock_store_after_evaluation	Restores normal operation by releasing _blocked and resolving _block_event.
add_successors	Adds one or more downstream GenericElement instances to the successors collection.
planned_stop_checker	Determines whether the asset is currently within a planned-stop interval.

Table 14 Essential attributes of Machine.

Name	Category	Description
required_capacity_to_start	R	Defines the minimum occupancy required before processing may begin.
start_event	C	Event used to trigger the start of processing once the minimum occupancy requirement has been satisfied.

Table 15 Essential methods of Machine.

Name	Description
_genericelement_store	Stores a reference to the store method inherited from GenericElement.
store	Extends the inherited store method by triggering start_event whenever the batching conditions are satisfied.
process	Defines the processing behaviour of the class.

Table 16 Essential attributes of AsyncMachine.

Name	Category	Description
waiting_arrival_events	C	Collection of events used to notify currently processing Material objects that a new Material object has entered processing and synchronisation may be required.

Table 17 Essential methods of AsyncMachine.

Name	Description
_arrival_alert	Triggers the alert mechanism stored in waiting_arrival_events.
_genericelement_store	Stores a reference to the store method inherited from GenericElement.
store	Extends the inherited store method by invoking _arrival_alert.
handle_post_yield_event	Handles events occurring after a yield operation, including random stops triggered by accumulated net processing time and the transfer of processed Material objects to downstream assets.
process	Defines the mechanism that allows Material objects to begin processing.
process_item	Handles the individual progression of each Material object's processing.

Table 18 Essential attributes of Conveyor.

Name	Category	Description
conveyor_length	R	Defines the total length of the conveyor. The value is dimensionless and should therefore use the same unit system as the dimensional attributes defined in Material.
conveyor_speed	R	Defines the speed at which Material objects move through the conveyor. The value is dimensionless and should therefore use the same unit system as the dimensional attributes defined in Material.
minimum_gap	R	Defines the minimum distance that must be maintained between consecutive Material objects. The value is dimensionless and should therefore use the same unit system as the dimensional attributes defined in Material.
entry_unique_token	C	Internal mechanism used to prevent race conditions when multiple Material objects attempt to enter the conveyor simultaneously.
wip	C	Internal structure describing all Material objects currently being processed within the conveyor.
waiting_for_space	C	Stores a Material object currently waiting for sufficient conveyor space to become available.
Material. length_to_time_conversion	C	Not an attribute of Conveyor. Instead, the conveyor dynamically injects this attribute into entering Material objects. The attribute represents the minimum remaining processing time required for the most recently admitted Material object to allow the entry of a new Material.

sync_events	C	Collection of synchronisation events associated with the Material objects currently under processing. Used to prevent race conditions.
enough_space_event	C	Event used when Material.length_to_time_conversion is lower than the remaining processing time of the most recently admitted Material object. It is used to coordinate the admission of new Material objects while maintaining the required spatial constraints.
spatial_attribute	R	Defines the name of the dimensional attribute in Material that represents the dimension travelling along the conveyor.
_get_spatial	C	Cached attribute getter used to improve performance when accessing the attribute specified by spatial_attribute.
item_fully_in_event	C	Event indicating that a Material object has completely entered the conveyor.

Table 19 Essential methods of Conveyor.

Name	Description
_asyncmachine_store	Stores a reference to the store method inherited from AsyncMachine.
store	Extends the inherited store method to include the geometric requirements associated with conveyor entry.
block_until_item_fully_in	Prevents unnecessary event handling when multiple Material objects attempt to enter the conveyor simultaneously. This is achieved through the creation of item_fully_in_event.
unblock_until_item_fully_in	Resolves item_fully_in_event, allowing waiting operations to continue.

calculate_yield_time	Extends the inherited implementation with synchronisation mechanisms required for simultaneous processing of multiple Material objects.
check_yielding_constraints	Extends the inherited implementation with synchronisation mechanisms required for simultaneous processing of multiple Material objects.
handle_post_yield_event	Extends the implementation inherited from AsyncMachine to include handling of geometric constraints.
process_item	Defines the processing behaviour of individual Material objects within the conveyor and coordinates the associated synchronisation mechanisms.

Table 20 Essential attributes of Buffer.

Name	Category	Description
minimum_send_size	R	Defines the minimum number of Material objects that must be accumulated before send_away can be initiated.
force_send_attribute	O	Optional rule requiring all stored Material objects to share a common attribute value. If an incoming Material object possesses a different value for the specified attribute, the currently stored Material objects are sent downstream even if minimum_send_size has not been reached.
_get_sendattr	C	Cached attribute getter used to improve performance when accessing the attribute specified by force_send_attribute.
items	C	Stores the Material objects currently contained within the buffer.

Table 21 Essential methods of Buffer.

Name	Description

process	The implementation depends on whether force_send_attribute has been defined. The method evaluates the buffer constraints and initiates the send_away process whenever the required conditions are satisfied.

Table 22 Essential attributes of MaterialArrival.

Name	Category	Description
batch_size_arrival	R	Defines the number of Material objects created during each arrival event.
product_mix_distribution	R	Placeholder attribute indicating that subclasses must define the possible arriving Material types and their associated probabilities.
product_mix_descriptors	R	Placeholder attribute indicating that subclasses must define how each possible Material type should be instantiated.

Table 23 Essential methods of MaterialArrival.

Name	Description
mix_generator	Abstract method that must be implemented by subclasses. Responsible for creating arriving Material objects, typically making use of product_mix_distribution and product_mix_descriptors.
process	Handles the arrival of Material objects, considering current occupancy levels, potential downstream constraints, waiting times between arrivals, and the transfer of newly generated Material objects to downstream assets.
arrival_time_calculator	Calculates the time between consecutive arrival events, including the effects of planned stops where applicable.

Appendix B: Description of specific simulation assets

Table 24 Essential attributes of Log.

Name	Category	Description
length	R	Attribute used for classification and relevant for conveyors where length represents the travelling dimension.
diameter	R	Attribute used for classification.

Table 25 Essential attributes of Plank.

Name	Category	Description
length	R	Attribute used for classification and relevant for conveyors where length represents the travelling dimension.
width	R	Attribute used for classification and may represent the travelling dimension in specific conveyor configurations.
thickness	R	Attribute used for classification and relevant for drying operations.
pallet_category	C	Derived from type_id. Defines the category of Pallet that should be formed using this type of Plank.

Table 26 Essential attributes of Pallet.

Name	Category	Description
thickness	R	Attribute used for classification and relevant for drying operations.
pallet_items	R	Stores references to all Material objects grouped within the pallet.

Table 27 Essential attributes of ExcelLogger.

Name	Category	Description

columns	C	Static attribute defined during initialisation. Defines the header columns used in the .xlsx log.
classifier	C	Static attribute defined during initialisation. Classifies statuses as active or inactive. Relevant for the .xlsx log.
colours	C	Static attribute defined during initialisation. Defines the colours associated with different statuses in the .xlsx log.
_data	C	Stores all logged data generated during simulation execution.
target_file	R	Directory where generated logs are written.
status_class	C	Static attribute defined during initialisation. Defines keywords used to differentiate machine statuses in the .csv log.
recognised_statuses	C	Static attribute defined during initialisation. Defines the possible statuses that each machine may assume. Relevant for the .csv log.

Table 28 Essential methods of ExcelLogger.

Name	Description
write_log	Handles the insertion of log entries into _data while maintaining the required ordering.
finalise	Generates the final .xlsx and .csv production logs after simulation execution has completed.
_sanitize_sheet_name	Removes or replaces characters that are not permitted in spreadsheet sheet names.
_drop_sender_production_pairs	Eliminates redundant starvation log entries generated when a machine returns to production immediately after a Material object leaves it.
_compute_durations	Calculates the duration of each log entry for presentation in the .xlsx log.

_compute_minuted_variables	Transforms the production log into the machine-learning format, where each row represents one minute and includes the relevant simulation variables.
_to_csv_chunks	Splits large .csv logs into smaller chunks to facilitate writing. The chunks are later merged into a single file.

Table 29 Essential attributes of LogArrival.

Name	Category	Description
random_uniform	O+C	Stores a RandomUniform random-number generator bounded between 0 and 1. An external random-number generator may be provided to control its behaviour.
prob_keys	C	Given an input of type dict[str, float] representing the probability of generating each Log type, this attribute stores the corresponding dictionary keys.
prob_cumulative	C	Given an input of type dict[str, float] representing the probability of generating each Log type, this attribute stores the cumulative probabilities associated with each category.

Table 30 Essential methods of LogArrival.

Name	Description
mix_generator	Generates Log objects according to the provided probability distribution and the standard attributes inherited from MaterialArrival.

Table 31 Essential attributes of SortingConveyor.

Name	Category	Description
length_per_typeid	R	Defines the travel distance associated with each Log classification.
process_time_distribution	C	Type hint: ConstantAttributeBased. Travel time is determined by length_per_typeid and conveyor_speed.

successors	C	Modified from a list into a dictionary to improve target selection during routing operations.

Table 32 Essential methods of SortingConveyor.

Name	Description
generate_process_time	Modified to determine travel time based on the classification of the incoming Log.
set_target	Modified to select a destination based on the attributes of the incoming Log.
add_successors	Modified to support dictionary-based successor management rather than list-based management.

Table 33 Essential attributes of TimeStamperBuffer.

Name	Category	Description
longest_time	R+C	Allows the user to define the residence time of the oldest stored Log at simulation time zero. The value is subsequently updated automatically during simulation execution.
master	C	Stores a reference to the MasterScheduler. The reference is assigned after both the scheduler and the TimeStamperBuffer objects have been instantiated.
_buffer_store	C	Stores a reference to the store method inherited from Buffer.

Table 34 Essential methods of TimeStamperBuffer.

Name	Description
store	Extends the inherited store method to update the residence-time tracking mechanisms whenever a Log enters the pile.
process	Reimplemented so that it is not continuously active within the simulation environment. Instead, it is attached to and detached from the environment on demand by the MasterScheduler, according to the current production schedule.

Table 35 Essential attributes of SawingConveyor.

Name	Category	Description
setup_time_distribution	R	Similar to process_time_distribution, consisting of a SimpyRNG used to generate setup durations.
new_setup_type	C	Indicates the setup type that is scheduled to be performed. Once setup is completed, this value becomes current_setup_type.
conveyor_speed	R+C	The inherited attribute is modified by the MasterScheduler when setup operations are performed, allowing conveyor speed to vary according to the active sawing pattern.
setup_block	C	Flag indicating whether the asset must enter the setup state.
undergoing_setup	C	Flag indicating whether a setup operation is currently being performed.
setup_block_event	C	Event used to coordinate waiting processes while setup is being performed.
current_setup_type	C	Indicates the currently active sawing pattern.
_conveyor_store	C	Stores a reference to the store method inherited from Conveyor.

Table 36 Essential methods of SawingConveyor.

Name	Description
store	Extends the inherited method to account for setup operations and the associated restrictions on material flow.
send_away	Reimplemented to handle the transformation of incoming Log objects into multiple outgoing Plank objects. Additionally, the method ensures that occupancy tracking remains consistent despite the change in object count.
setup_blocker	Invoked by the MasterScheduler to force the machine into a setup state, allowing the active sawing pattern to be changed.

setup_unblocker	Restores normal operation after setup has been completed and allows new Material objects to enter the asset.
setup	Performs the setup operation associated with changing the active sawing pattern.

Table 37 Essential attributes of PalletStacker.

Name	Category	Description
maximum_send_size	R	Defines the maximum number of Plank objects that may be grouped into a single pallet.
current_pallet_type	C	Tracks the pallet category currently being assembled.
successors	C	Modified from a list into a dictionary to support multiple downstream destinations.
_buffer_send_away	C	Stores a reference to the send_away method inherited from Buffer.

Table 38 Essential methods of PalletStacker.

Name	Description
send_away	Modified to create Pallet objects from the accumulated Plank objects. References to all contained Plank objects are stored in the pallet_items attribute of the newly created pallet.
process	Extends the inherited implementation to consider the maximum_send_size constraint.
add_successors	Modified to support dictionary-based successor management.
set_target	Modified to select a destination based on the pallet_category associated with the contained Plank objects.

Table 39 Essential attributes of PalletBuffer.

Name	Category	Description
target	C	Defines the Dryer currently designated to receive Pallet objects. The value is continuously updated by the DryerHandler.

ready_to_send_items	C	Similar to items_to_produce, but updated at a different stage of the simulation-event cycle. This mechanism is used to prevent race conditions.
master	C	Stores a reference to the DryerHandler. The reference is assigned after both the handler and the PalletBuffer objects have been instantiated.
_buffer_store	C	Stores a reference to the store method inherited from Buffer.

Table 40 Essential methods of PalletBuffer.

Name	Description
store	Modified to update ready_to_send_items rather than relying solely on the inherited behaviour.
process	Reimplemented so that it is activated and deactivated on demand by the DryerHandler rather than running continuously within the simulation environment.
set_target	Modified to return the value stored in the target attribute.

Table 41 Essential attributes of Dryer.

Name	Category	Description
process_time_per_type	R	Similar to process_time_distribution, but uses the type hint dict[str, SimpyRNG] to define processing times for different pallet categories.
loading_planned_stop_loop_length	O	Equivalent to planned_stop_loop_length, but associated with the loading and unloading schedule of the drying chambers.
loading_planned_stops	O	Equivalent to planned_stops, but associated with the loading and unloading schedule of the drying chambers.

_loading_planned_intervals	C	Equivalent to _planned_intervals, but associated with the loading and unloading schedule.
_loading_planned_starts	C	Equivalent to _planned_starts, but associated with the loading and unloading schedule.
master	C	Stores a reference to the DryerHandler. The reference is assigned after both the handler and the Dryer objects have been instantiated.
.master_index	C	Represents the position occupied by the dryer within the collection of dryers maintained by the DryerHandler.
_machine_send_away	C	Stores a reference to the send_away method inherited from Machine.
_machine_store	C	Stores a reference to the store method inherited from Machine.

Table 42 Essential methods of Dryer.

Name	Description
store	Extends the inherited implementation to incorporate loading and unloading schedules.
send_away	Extends the inherited implementation to incorporate loading and unloading schedules.
generate_process_time	Modified to generate processing times according to Pallet.pallet_category.
loading_stop_checker	Equivalent to planned_stop_checker, but uses the loading-schedule attributes instead of the standard planned-stop attributes.

Table 43 Essential attributes of Depalletiser.

Name	Category	Description
_buffer_store	C	Stores a reference to the store method inherited from Buffer.

Table 44 Essential methods of Depalletiser.

Name	Description
store	Modified so that incoming Pallet objects are immediately disassembled and their contained Plank objects are introduced individually into the production flow.

Table 45 Essential attributes of InspectionConveyor.

Name	Category	Description
successors	C	Modified from a list into a dictionary to support multiple downstream destinations.

Table 46 Essential methods of InspectionConveyor.

Name	Description
add_successors	Modified to support dictionary-based successor management.
set_target	Modified to select a destination according to the type_id of the processed Plank object.

Table 47 Essential attributes of MasterScheduler.

Name	Category	Description
env	R	Stores a reference to the simulation environment.
sawing_conveyor	R	Stores a reference to the SawingConveyor.
stacks	R	Stores references to the TimeStamperBuffer objects representing the different log piles. The buffers are organised as a dictionary whose keys correspond to log categories.
setup_types	R	Stores information linking log categories to plank categories, including the number

		of produced planks per log and the associated processing speed of the SawingConveyor.
time_reference_multiplier	C	Converts age constraints expressed in days into the internal time unit used by MeasuredEnvironment.
max_time_in_stacks	R+C	Defines the maximum residence time permitted for logs within the storage piles. The scheduler uses this value when assigning processing priorities.
max_time_tolerance	R+C	Defines a tolerance interval associated with max_time_in_stacks. Log piles whose oldest element exceeds the maximum residence time minus this tolerance receive the highest priority.
_avoid_excessive_calls	C	Internal mechanism used to limit the frequency of age updates, preventing excessive event generation within the simulation.
opened_stack	C	Stores a reference to the log pile currently supplying material to the SawingConveyor.
last_update_time	C	Stores the most recent simulation time at which log-age information was updated.
no_logs	C	Flag indicating that all log piles are empty. This enables TimeStamperBuffer objects to notify the scheduler when new Log objects arrive.

Table 48 Essential methods of MasterScheduler.

Name	Description
open_stacks	Communicates to the TimeStamperBuffer objects which pile should begin supplying material to the SawingConveyor.
generate_new_order	Determines which TimeStamperBuffer should be processed next.

update_time_trackers	Updates the residence-time information of the different TimeStamperBuffer objects based on the age of their oldest stored Log.

Table 49 Essential attributes of DryerHandler.

Name	Category	Description
env	R	Stores a reference to the simulation environment.
wait_event	C	Event used to prevent race conditions when multiple PalletBuffer and Dryer objects invoke evaluate_new_order simultaneously.
dryers	R	Stores references to the available Dryer objects. These are organised as a list of tuples containing the dryer reference, its minimum loading requirement, and an indicator specifying whether the dryer is currently empty.
pallet_stacks	R	Stores references to the PalletBuffer objects associated with the different pallet categories. The buffers are organised as a dictionary whose keys correspond to pallet categories.

Table 50 Essential methods of DryerHandler.

Name	Description
execute_new_order	Communicates a new production order to the relevant PalletBuffer and Dryer objects and updates the internal scheduling variables accordingly.
evaluate_new_order	Invoked whenever a Dryer becomes available or a PalletBuffer receives a new pallet. Evaluates whether a new production order should be generated and, if so, determines the most appropriate one.

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE
DIVISION OF PRODUCTION SYSTEMS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se