Event Log Analysis on Realistic Queue Qualification Tests

Event Log Analysis on Realistic Queue Qualification Tests: Developing a Graph Theory-based Diagnostic Tool

Faezeh Sattari

October 2023

Eindhoven University of Technology
Stan Ackermans Instituite – Software Technology

Partners

ASML Eindhoven University of Technology

Steering Group
(By alphabetical order)

Marc Hermans
Andre Korbes
Mari Mnatsakanyan
Junchao Xu

Date October 2023

Composition of the Thesis Evaluation Committee:

Chair:	Jacob Krüger
Members:	Marc Hermans
	Andre Korbes
	Jacob Krüger
	Mari Mnatsakanyan
	Junchao Xu

The design that is described in this report has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct.

Contact Address
Eindhoven University of Technology
Department of Mathematics and Computer Science
MF 5.072, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
+31 402474334

Partnership
This project was supported by Eindhoven University of Technology and ASML.

Published by
Eindhoven University of Technology
Stan Ackermans Instituite

EngD-report
PDEng REPORT NUMBER

Preferred reference
Improve feedback loop: Intelligent diagnostics of realistic queue qualification test. Eindhoven University of Technology, EngD Technical Report PDEng REPORT NUMBER, October 2023

Abstract
The Metrology department at ASML develops algorithms that drive most of the parts within lithography scanners to achieve performance. Those algorithms use parameters that are set by calibration software during machine setup.
To provide a feedback loop for Metrology software development, an automated software tester at the system level is used to qualify deliveries of metrology driver software and calibration software. This tester executes a sequence of calibration tests (namely CPDs) according to a predefined setup sequence and verifies the setup's final performance.
Metrology drives constantly for improving efficiency in software development and strives to automate processes. In this context, we would like to have tools (algorithms) that analyze data and generate reports about the results and issues automatically. This tool would be used to automate the diagnosis process and decrease the investigation time to find the hints to the root cause.

Keywords
Metrology, Lithography, Realistic queue, Test results, Automation, Analysis, Investigation

Foreword

In ASML, analyzing event logs from the machine is part of everyday activities for D&E. In particular, these are event logs from hundreds of automated tests that are used for regression testing. To reduce the time the engineers spend analyzing these test results, we aim to create an intelligent system that will automatically identify the possible root cause of the failure.

Faezeh has created a good basis for developing such a system. In this thesis, Faezeh shows that a graph theory-based method is well suited for the purpose of analyzing the event logs from ASML machines. To prove that the selected method is able to provide accurate results, Faezeh created a tool that now can be used by the ASML engineers to identify the root of the software failure more easily. More importantly, the solution that Faezeh offers has great potential for growing into an intelligent system that will automatically provide comprehensive feedback on the test results. Faezeh also suggests a way forward towards this goal and identifies possible issues in her work.

Mari Mnatsakanyan
October 2023

Preface

This document presents the final report of the "Event Log Analysis on Realistic Queue Qualification Tests" project, conducted by Faezeh Sattari as part of the EngD program in Software Technology at Eindhoven University of Technology. The project, in collaboration with ASML, aimed to automate failure diagnosis in Realistic Queues tests, reducing investigation time for developers.

The tool utilizes algorithms to analyze data, generate reports, and prototype graph theory techniques for potential solutions. It includes solution description, architecture, design, verification, and validation. Chapters 2, 3, and 4 cover problem, domain, and requirements. Chapter 5 addresses project management, risks, and mitigation. Chapter 6 delves into the architectural logic, while implementation details are discussed there too. Chapter 7 covers verification, validation, and results. Chapter 8 offers project achievements and future suggestions.

Faezeh Sattari
October 2023

Acknowledgements

I wish to extend my sincere gratitude to the collaborative efforts of both the ASML and TU/e, whose collective contributions have been instrumental in the successful completion of this investigation. I am deeply appreciative of the guidance provided by ASML supervisors Mari Mnatsakanyan, Andre Korbes, and Junchao Xu. While Mari and Andre's dedication in aligning the system design with stakeholders' concerns and their invaluable support in navigating intricate challenges within ASML have been pivotal in shaping the trajectory of this research, Juancho’s insights and supervision have also been invaluable to the project's progress.

Furthermore, I would like to acknowledge Jacob Kruger, my supervisor from TU/e, whose consistent oversight of my progress and timely guidance have significantly contributed to the project's accomplishments.

Within ASML, I am thankful for the valuable insights and support received from colleagues across various departments. The expertise and guidance provided by Marc Hermans, ASML Group Leader, regarding project success and adherence to company regulations, have greatly enriched the project's outcomes. Jenny van Baden, ASML Software Field Performance Engineer, has been instrumental in emphasizing the significance of visualization effects and manual inspection within the analytics and visualization domain. Andreas Gammel, ASML Engineer, has provided valuable contributions in the verification of graph theory-based methods and pollution detection techniques. Erdem Karayer's insights into software development technical limitations and root cause automation tools, coupled with Errol Zalmijn's expertise in data science techniques, have been indispensable.

October 2023

Executive Summary

This report outlines the design and implementation of an automated tool to analyze realistic queue event logs at ASML, a premier chip-making equipment manufacturer. The project's core objective was to create a diagnostic tool capable of identifying the root causes of issues that arise during the execution of the realistic queue within ASML's Metrology department. Employing graph theory-based methods, the developed system efficiently detects root causes within event logs. These methods are fast, memory-efficient, and require no retraining, rendering them an ideal solution for ASML's dynamic operational context.

The realistic queue plays a pivotal role in ASML's lithography machines, generating substantial data that offers crucial insights into system behavior. Given the data's volume, an automated analytical approach is imperative. The system designed for this project employs algorithms to analyze event logs generated by the realistic queue, producing comprehensive reports on outcomes and identified issues.

To achieve project goals, a range of requirements were collected, guiding architectural decisions on method selection. The resulting designed tool boasts remarkable scalability, adeptly handling large data volumes within tight time and memory constraints. Moreover, we designed this tool to integrate with the existing realistic queue tester and harmonize effectively with cloud-based architecture in ASML.

Efficiency optimization emerged as a primary challenge during the project. Ensuring prompt feedback to Metrology software developers was a critical aspect, alongside managing the order of linked events as defined by developers and filtering the pollution event log list. The system triumphantly overcame these challenges and operates within defined time and memory limitations.

In conclusion, we developed the tool to offer a potent solution for automating the analysis of realistic queue event logs within ASML's Metrology department. Boasting optimization for speed and efficiency, the system incorporates features ensuring the correct event order and effective filtering of the pollution event logs list. Overall, we could enhance the efficiency of event log analysis diagnostics for realistic queue qualification tests.

List of Figures

List of Tables

Introduction

In today's highly interconnected and data-driven world, it is becoming increasingly important for organizations to efficiently process and analyze the large amounts of data generated by their systems.[1] This is especially true in the high-tech industry, where complex and sophisticated systems generate vast amounts of data that can provide insights into system behavior and performance.

One such industry leader is ASML, a leading manufacturer of chip-making equipment and a key player in the semiconductor industry. ASML's Metrology department develops algorithms that drive most of the parts within lithography scanners to achieve performance. To provide a feedback loop for Metrology software development, an automated software tester at a system level is used to qualify deliveries of metrology driver software and calibration software. In this context, the ability to automatically analyze the event logs generated by ASML's realistic queue tester is crucial for improving the efficiency of the software development process.

ASML is a company that produces complex machines, such as the Twinscan NXT and NXE machines, which are used in semiconductor manufacturing. These machines have multiple components that are responsible for various functions, and any failure in these components can cause the entire machine to malfunction. There could be various reasons for system breakdown, such as inappropriate working environments, logistic errors, or operator errors.

Due to the high costs and risks involved in testing on a real machine, ASML uses a virtual machine called Simulated test environment to test software, and all the activity events are continuously recorded in event logs. These logs not only provide information about when the system works correctly but also indicate when and where problems occur. However, diagnosing the root cause of failure from these event logs can be a challenging task. The time used in diagnosis is usually much longer than the time used in problem-solving because event logs often contain thousands of events, and only a few of them are important records relevant to the root cause of the failure.

ASML developed some tools to detect anomalies, but it's difficult to definitively judge its accuracy. It is indeed used extensively for customer machine data analysis. However, it's limited in real queue tests, where manual use and event log conversion are necessary.

While ASML inhouse tools excels in troubleshooting recurrent errors, our project primarily focuses on realistic queue scenarios, without a direct comparison to other tools' accuracy.

1.1 Goals and Objectives

The overarching goal of this thesis was to design a tool capable of analyzing Simulated test environment data and utilizing advanced algorithms to pinpoint the root causes of failures or anomalies within the results report.

In line with this aim, the research set out to achieve the following specific objectives:

- **Objective 1: Eliciting Requirements and Informing Architectural Decisions**

  This objective centered on the precise identification of system requirements and the informed selection of architectural decisions. By doing so, the aim was to ensure a seamless alignment of the system with the intended functionalities and performance benchmarks.

- **Objective 2: Designing and Implementing an Automated Queue Event Log Analysis System**

The second objective involved the design and successful implementation of a resilient and efficient system. This system was tasked with automating the comprehensive analysis of extensive queue event logs. This automation not only streamlined the process but also significantly reduced the need for manual intervention. Consequently, the outcome was a marked reduction in errors, coupled with a notable improvement in the overall accuracy of the analysis.

By attaining these objectives, this research has laid the groundwork for an automated solution that holds the potential to greatly enhance the efficiency and precision of anomaly detection through the automated analysis of realistic queue event logs.

In summary, this research aims to deliver an automated system that enhances efficiency, accuracy, and scalability in anomaly detection. By achieving these objectives, developers will have a powerful tool for automating queue analysis, reducing manual effort, and making informed decisions to optimize queue performance in real-world scenarios, including tests conducted in a Google Cloud environment.

The proposed tool will provide a smarter and more accurate diagnostic approach to help engineers save time and improve the efficiency of problem-solving in ASML's machines. The tool's ability to visualize events and results will enhance the understanding of the root cause of failures or abnormalities and provide an actionable report to engineers. The use of state-of-the-art algorithms will ensure accuracy and reliability in the root cause analysis of event logs.

1.2 Outline

This report consists of several chapters, each serving a specific purpose:

- Chapter 1: Introduction - Outlines objectives and report structure.
- Chapter 2: Problem Analysis - Conducts a thorough problem analysis and defines scope and goals.
- Chapter 3: Project Domain - Explores the project's domain, including ASML and related terminology.
- Chapter 4: Requirements and Use Cases - Focuses on requirements, including business, functional, and quality attributes.
- Chapter 5: Feasibility Assessment - Assesses project feasibility and addresses potential issues.
- Chapters 6 and 7: Design and Implementation - Provide insights into the intelligent diagnosis tool.
- Chapter 8: Conclusion - Presents findings, recommendations, and reflections on company contributions.

Problem Analysis

In this chapter, we discuss the project context and undertake a problem analysis for the automation of event log diagnosis. The primary objective of this project is to design and implement a system capable of automating the analysis of realistic queue event logs, generating reports on the results and issues automatically. The system is expected to integrate seamlessly with the realistic queue tester and be compatible with cloud infrastructure.

2.1 Problem Definition

The ASML system generates a large amount of data from the realistic queue tester, which is difficult to analyze manually. The current system lacks the necessary tools and algorithms to analyze and generate reports automatically. This manual process is time-consuming and error-prone, resulting in a delay in identifying the root cause of the issues. Therefore, the problem is to design and implement a system that can automate the analysis of realistic queue event logs and generate reports about the results and issues automatically. The system should also integrate with the realistic queue tester and be cloud compatible.

2.2 Project Goal

The goal of this project is to design and implement a system that can automate the analysis of realistic queue event logs and generate reports about the results and issues automatically. The system should integrate with the realistic queue tester and be cloud compatible. The system should provide intelligent diagnostics for the realistic queue tester and analyze the data generated by the ASML machines. Additionally, the project aims to gather the requirements and perform architectural decisions for the system's design and implementation.

Based on the project goals, the aim of this study is to design and implement a system that automates the analysis of realistic queue event logs and generates reports about the results and issues automatically. The system is required to integrate with the realistic queue tester and be cloud compatible. Additionally, the system should provide intelligent diagnostics for the realistic queue tester and analyze the data generated by the ASML machines.

To achieve these goals, I identified several key requirements for the system. Firstly, the system should have tools and algorithms that can effectively analyze data from the tester, enabling it to generate reports about the results and issues automatically. Secondly, the system should be capable of visualizing results and events, making it easier for stakeholders to understand the data generated by the system.

Another crucial requirement is the ability of the system to indicate the root cause of failures/abnormalities in the results report. This will enable users to quickly identify and address issues, reducing the time and resources required for troubleshooting. Additionally, the system should provide intelligent diagnostics for the realistic queue tester, analyzing the data generated by the ASML machines and providing useful insights to stakeholders.

To achieve these requirements, the study involves investigating, designing, and implementing software that provides intelligent diagnostics for the realistic queue tester. I also provided the analysis and definition of functionalities of the diagnostics tool. Additionally, I added requirement gathering documentation to ensure that the system meets the needs of stakeholders.

Furthermore, the study evaluates existing software tools, libraries, visualization, data science, and machine learning techniques to make informed architectural decisions. The integration of the system with

the realistic queue tester is to ensure that the diagnostics are executed as part of the tester at the end. Finally, the system is designed to be cloud compatible as the realistic queue tester needs to be executed on the cloud.

Overall, this study aims to provide an automated system that can effectively analyze realistic queue event logs and provide useful insights to stakeholders, reducing the time and resources required for troubleshooting. The system will provide a valuable tool for the ASML machines, allowing them to generate reports and analyze data more effectively.

2.3 Methodology

This section presents the methodology used for the problem analysis of the automation of the event log diagnosis project. The objective is to identify and understand the key entities involved in automating the diagnosis of event logs, with a particular emphasis on the core functionality of an anomaly detection system. The methodology consists of the following steps:

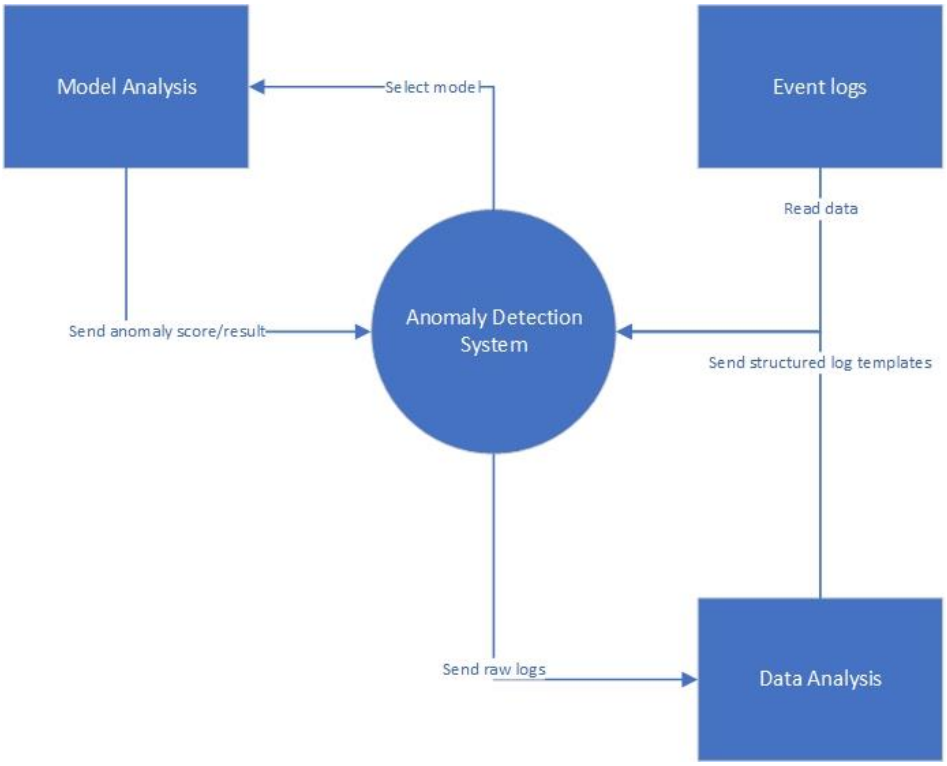

Figure 1 The data flow diagram of the anomaly detection system

Figure 1 shows the context diagram created to visualize the overall system and its entities. The central entity of the system is the anomaly detection system, which forms the core component of the analysis process. The context diagram illustrates the relationships and interactions between the anomaly detection system and other entities relevant to the project. These entities include:

a. Event Logs:

Event logs serve as the primary input for the diagnosis process. They contain recorded events and relevant data that require analysis.

b. Model Analysis:

Multiple analysis models are employed to extract meaningful insights from the event logs. These models may include process mining, statistical analysis, machine learning, and pattern recognition techniques. Each model contributes to different aspects of the diagnosis process.

c. Data Analysis:

Developed modules for parsing and pre-processing the event logs, extracting pertinent information, and preparing the data for analysis. These modularities could encompass log parsing libraries, software applications, or custom scripts.

By following this methodology, a comprehensive understanding of the entities involved in automating event log diagnosis can be achieved. The problem analysis phase provides insights into the suitability and effectiveness of the anomaly detection system, leading to valuable improvements in the automation process.

Note: The specific analysis techniques, tools, and approaches used within each entity may vary depending on the project's requirements and the nature of the event logs being analysed.

2.4 Main Questions

The main questions help to define the project's objectives and requirements. The four main questions for this project are:[2]

- What do you want the system to do?

The system should automate the analysis of realistic queue event logs and generate reports on the results and any issues detected. It should also visualize the results and events and indicate the root cause of failures or abnormalities.

- What is the purpose of the system you want?

The primary objective of this system is to establish a fast feedback loop in the realistic queue tester environment. Its core purpose is to expedite the diagnostic process, ensuring swift identification of issues, and subsequently reduce investigation time significantly. Moreover, it aims to enhance the overall efficiency and accuracy of the testing process.

- What do you want to be able to do?

The stakeholders want to be able to analyze the results of the realistic queue tester automatically, generate reports on the results and issues, and visualize the results and events. They also want to be able to investigate and diagnose any failures or abnormalities detected during testing.

- Who should be asked?

The stakeholders who should be asked include the project sponsor, project manager, technical team, testing team, and end-users.

2.5 Stakeholder Analysis

To identify potential stakeholders for the project, I attended group meetings and networked with experts in various departments of ASML, including the EUV Metrology Department and the Test-related Activities Department. These meetings provided me with valuable insights into the potential pitfalls of different analysis methods, which helped me to narrow down my approach to the project. Through these meetings, I also connected with a highly knowledgeable individual who was experienced in this area and provided valuable input on the project's scope and methodology. In addition to these meetings, I also reviewed similar thesis projects that had been completed by master's students in the past, which helped me to identify other stakeholders who had previously worked on similar projects.

Stakeholder analysis is a critical step in any project, and it is especially important when working on a complex project like this. It involves identifying all parties that will be affected by the project and understanding their concerns, requirements, and levels of involvement. This information is used to develop strategies for managing stakeholders and ensuring that their needs are met throughout the project lifecycle.

In this project, the primary stakeholders are ASML Netherlands B.V. and the Eindhoven University of Technology. However, there are many other stakeholders who may be impacted by the project, including employees in the EUV metrology and test-related activities departments, software developers, and end-users of the diagnostic tool. By conducting a thorough stakeholder analysis, we can identify potential conflicts and develop strategies for managing them, ultimately increasing the likelihood of project success.

Once I had identified the stakeholders, I developed a plan for engaging with them and understanding their requirements and concerns. This involved conducting interviews, surveys, and focus groups to gather feedback and input on the project.

In addition to identifying the stakeholders, it is also important to analyze their requirements and constraints. This involves understanding their goals, concerns, and expectations for the project and identifying any factors that may limit or impact their involvement. By doing so, we can develop strategies for managing the stakeholders and ensuring that their needs are met throughout the project lifecycle.

Overall, stakeholder analysis is crucial for project success. It involves identifying, engaging, and managing stakeholders throughout the project. The ultimate goal is to ensure that all stakeholders accept the project's outcome, which defines its success.

2.6 Usage Model

The usage model defines the scope and out-of-scope elements of the system, as well as the overall scenario and capabilities derived from the scenario. In this project, the scope includes the automated analysis of realistic queue event logs, report generation, visualization, and root cause analysis. The out of scope elements include the development of the realistic queue tester itself.

The overall scenario involves the testing of the realistic queue using the tester and the automated analysis of the event logs. The capabilities derived from the scenario include the ability to diagnose failures or abnormalities, generate reports, and visualize results and events.

Scope and Out-of-scope Elements

Defining the scope and not-scope of a system is an important step in the methodology of any project. It helps to clarify what the system is supposed to do and what it is not supposed to do. The scope and not-

scope of the system provide the boundary for the project, and all requirements and use cases should be within the scope.

The scope of the system for this project includes designing and implementing a system to automate the analysis of realistic queue event logs, integrating the system with a realistic queue tester, and designing a cloud-compatible system. The system should be able to analyze data from the tester, generate reports automatically, visualize results and events, indicate the root cause of failures/abnormalities in the results report, and provide intelligent diagnostics for the realistic queue tester.

The not-scope of the system includes anything that is outside of the above-defined scope. For example, the system is not responsible for the maintenance or repair of the tester hardware or software. It is not responsible for monitoring the tester or controlling its operations. The system is also not responsible for providing training on the tester or its usage. The not-scope should be clearly defined to avoid misunderstandings and scope creep during the project.

Overall Scenario

In the overall scenario, the system will automate the analysis of realistic queue event logs generated by the ASML machines using a cloud-compatible solution. The system will provide intelligent diagnostics and generate reports about the results, issues, and root causes of failures/abnormalities in the results report. The main objective of the system is to improve the efficiency of the ASML machines and reduce the overall downtime by providing accurate and timely insights into the root causes of issues.

To achieve this objective, I designed the system to collect and preprocess data from the ASML machines and perform log parsing to extract relevant information from the event logs. The system uses advanced algorithms and data science techniques to analyze the data and identify patterns and anomalies that could indicate issues or abnormalities in the machine's performance.

The system provides a user-friendly interface to visualize the results and events, allowing the stakeholders to quickly identify potential issues and take appropriate actions. We integrated the system with the Realistic Queue tester, and the diagnostics are executed as part of the tester at the end, providing a seamless experience for the users.

Overall, the system will enable the stakeholders to make data-driven decisions, improving the efficiency and performance of the ASML machines and reducing the downtime, leading to significant cost savings for the company.

Deriving Capabilities from The Scenario

Deriving capabilities from the scenario involves identifying the key features and functions that the system must have to support the identified use cases. This includes breaking down the use case scenarios into specific tasks and activities, and then identifying the system capabilities required to support those tasks and activities.

For example, in the scenario outlined in section 2.6.2, the system must possess the following functionalities:

1. Parse queue event logs and extract relevant information
2. Analyze extracted information using algorithms and tools
3. Generate automated reports on analysis results and issues
4. Visualize results and events in an intuitive and clear manner
5. Identify root causes of failures/abnormalities in the results report
6. Provide intelligent diagnostics for the realistic queue tester

7. Support cloud compatibility for executing the realistic queue tester

By identifying these capabilities, the project team can better define the requirements for the system and begin to design and implement the necessary functionality to support these capabilities.

2.7 Conclusion

In summary, this chapter focuses on the problem analysis phase of the project, which aims to design and implement a system for automating the analysis of realistic queue event logs. The identified goals include generating automated reports, integrating with the realistic queue tester, being cloud compatible, and providing intelligent diagnostics for the tester and ASML machine data. The requirements gathering process is discussed, highlighting the need for effective data analysis, and root cause identification.

Domain Analysis

In this chapter, I am going to break down a crucial step: understanding the area where we're applying automation to the event log diagnosis project. Specifically, we are going to zoom in on the ASML system, focusing on the part that directly relates to our project. This helps us get familiar with the special words and phrases used in this field. This deep dive sets us up to build a strong base for the next stages of the project. We also discuss the types of data involved, particularly event logs, and provide an overview of the ASML testing system and diagnosis tools. Furthermore, we examine related works outside the ASML domain to gather insights and leverage existing knowledge. Through domain analysis, we aim to gain a comprehensive understanding of the project context and domain-specific aspects, laying a solid foundation for the subsequent stages of our research.

3.1 ASML

ASML is a key player in the chip industry, making microchips vital to modern tech like smartphones, cars, and medical devices. They've been improving lithography tech since 1984, offering full solutions for making complex patterns on silicon wafers. They partner with industry leaders like Intel and Samsung to make critical machines for IC production [1].

ASML's EUV metrology department spearheads research and development, focusing on novel measurement techniques and tools to enhance lithography machine performance. They conduct vital tasks like testing, validating new methods, developing test plans, conducting experiments, analyzing data, and collaborating across departments to integrate findings into the design process.
The EUV metrology department's rigorous testing and analysis are pivotal in ensuring the reliability and performance of ASML's lithography machines, contributing to cost reduction and risk mitigation.
In summary, ASML's EUV metrology department plays a vital role in ASML's R&D efforts, contributing to the success of the company by ensuring high standards of performance and reliability.

3.2 Related Works

In this section, various approaches for event log analysis in industrial settings are summarized:

1. Rule-based mathematical methods are simple and suitable for small, straightforward systems but become unwieldy for complex setups.
2. LSTM, a deep learning technique, excels in handling sequential data like time-series but struggles with complex and interconnected events.
3. Graph theory solutions offer visual representations of event relationships, beneficial for identifying anomalies but may not suffice for complex, large-scale event logs without additional methods like clustering.

In conclusion, the choice of analysis method should align with the system's complexity and size. Rule-based methods suit simple systems, LSTM is effective for sequential data, and graph theory solutions provide visual insights but may require complementing approaches. Combining methods often yields a comprehensive event log analysis. Table 1 shows the tradeoff analysis between these methods. [4][6][9]

Table 1 Trade off analysis of related methods

Methods	Advantages	Disadvantages
rule-based mathematical methods	✓ Low complexity ✓ Simple implementation	➢ Not suitable for large scale complex systems
LSTM	✓ Sequential data ✓ Time-series ✓ Text	➢ Not appropriate for highly interconnected events ➢ LSTM is to model temporal dependencies between in sequences but not relationships between events ➢ Struggle to handle complex data ➢ Not appropriate for the models with relationships between individual events
Graph theory solutions	✓ Graph-theory can provide a visual representation of the relationships between events ✓ Help to identify unusual patterns or anomalies ✓ Low complexity ✓ Better representation ✓ Enhanced analysis	➢ Hard interpretation in large scale and highly interconnected event logs ➢ Not sufficient for complex and large-scale event logs(based on simple statistical methods) ➢ It needs to be combined with other methods like clustering

Implementing effective anomaly detection systems presents several challenges, including real-time capability, model retraining for false positives, diverse and representative dataset collection, and managing the associated data collection costs. Generalizing well on new data, preventing overfitting, and validating performance in various scenarios are also crucial. Additionally, having domain knowledge about metrology setup is important. Risks in implementation include compatibility issues with cloud systems, licensing concerns, and the need for administrative access on company laptops. The choice of anomaly detection methods varies depending on system requirements and event log types. Rule-based and statistical methods offer simplicity but may not be suitable for large-scale systems. LSTM handles sequential data but struggles with complex interconnected events. Graph theory solutions like SNA and GNN provide visual representations of event relationships but may lack complexity for extensive logs. Combining SNA and GNN can enhance analysis but at the cost of increased computational complexity and complex output interpretation.[8]

Requirements and Use Cases

In this chapter, we embark on a comprehensive journey to outline the requirements and use cases of our project, setting the stage for a systematic and purpose-driven development process. This pivotal step bridges the gap between stakeholder aspirations and tangible system functionalities.

The requirements were categorized using the MoSCoW method through a series of structured meetings for requirement registry. During these sessions, priority levels were carefully revised and finalized in close consultation with supervisors. The application of the MoSCoW method involved thoughtful consideration and consensus-building among project stakeholders to accurately classify requirements into Must have, Should have, Could have, and Won't have categories. The process was meticulous, involving detailed discussions, assessments, and iterative feedback loops to ensure precise assignment of these labels aligning with project objectives and stakeholder expectations.

4.1 Use Cases

In this section I present a detailed exploration of the system's functionalities as perceived and interacted with stakeholders. It defines specific actions or goals that a user can undertake when engaging with the system, encompassing data preprocessing, automated analysis, and result interpretation. The identified use cases provide a structured framework to understand the system's behavior, allowing users to effectively identify root causes of given issues.

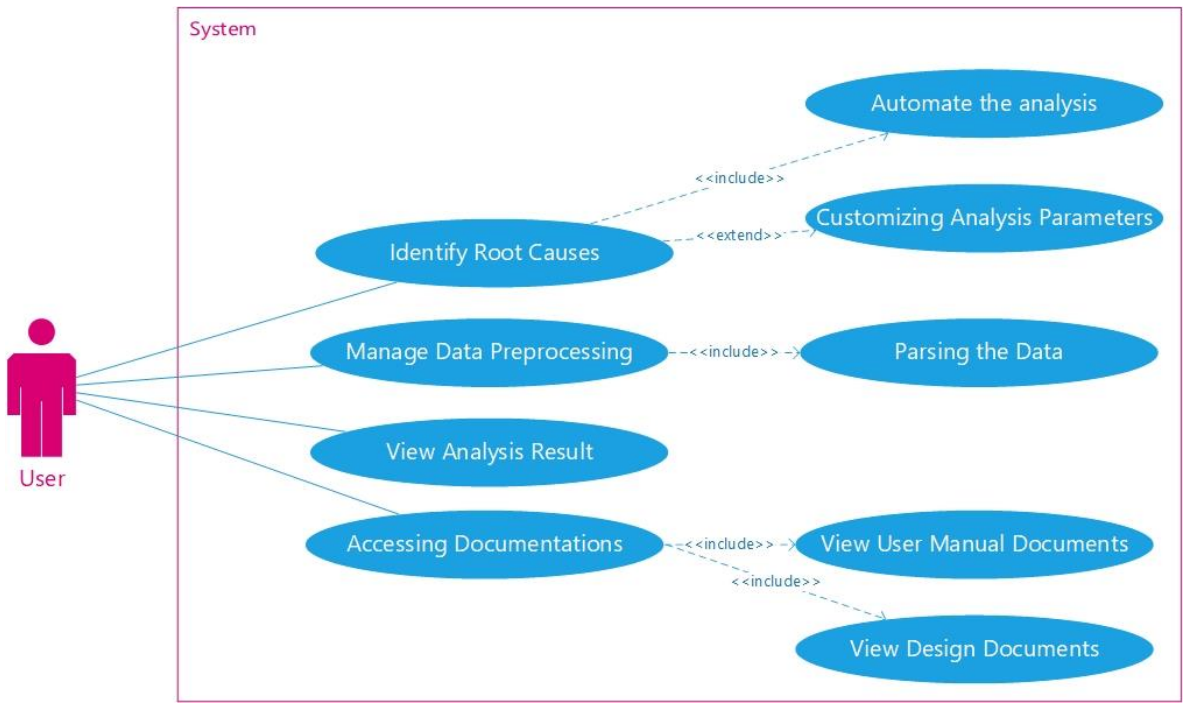

Figure 2 Use case diagram of the system

Based on Figure 2, I categorized the use cases into four key functionalities, each essential for effective system operation and management:

1. **Identify Root Causes:**
   Description: Identifies root causes of issues within the system.
   Includes: Automation of analysis processes.

Extends: Customization of analysis parameters.

2. **Managing Data Preprocessing:**
   Description: Prepares and parses data for further processing.
   Includes: Parsing raw data to enhance usability.

3. **View Analysis Result:**
   Description: Provides access to analysis results.

4. **Accessing Documentation:**
   Description: Facilitates access to crucial system documentation.
   Includes: Viewing user manual documents for operational insights.
   Includes: Viewing design documents for understanding system architecture.

4.2 Requirements Elicitation: A Collaborative Approach

The foundation of any successful project rests on a clear understanding of what needs to be achieved. Our requirements elicitation process was not merely a documentation exercise; it was a collaborative effort aimed at synergizing stakeholder insights and technical feasibility. Through proactive networking and a series of biweekly meetings, I engaged with a diverse range of stakeholders across ASML. These interactions fostered an environment for open discussions, enabling the identification of key priorities, expectations, and potential challenges.

Iterative Prototyping: To enhance the accuracy and relevance of requirements, an iterative prototyping approach was adopted. Prototypes were progressively developed and shared with stakeholders. This iterative process allowed us to refine requirements based on continuous feedback, ensuring alignment with stakeholder needs.

Based on the MoSCoW method, the requirements were categorized into business, functional, non-functional, architectural, and implementation requirements. Each category was meticulously shaped to cater to the diverse needs and aspirations of the project.

Business Requirements

Business requirements encapsulate the overarching goals and aspirations of our project. They serve as the beacon guiding the development process. Notable among these requirements are:

B01: System Event Log Collection and Storage: The system's ability to collect and store system event logs is essential for historical analysis and informed decision-making.

B02: Automated Event Log Analysis: The automation of event log analysis to identify potential root causes aligns with the project's objective of enhancing efficiency in issue resolution.

B03: Models and Algorithms: This requirement underscores the significance of robust models and algorithms as the backbone of accurate event log analysis.

B04: Seamless Integration: The capability to seamlessly integrate with the existing IT system is essential for harmonious operation within the ASML ecosystem.

B05: Handling Large Volumes: The system's capacity to manage extensive event logs and swiftly provide root cause analysis contributes to timely decision-making.

B06: User-Friendly Interface: The importance of a user-friendly interface cannot be understated. This requirement focuses on ensuring ease of use, fostering user engagement, and efficient information access.

This section lists the high-level statements of the goals, objectives, and requirements that should be met. The business requirements of this project are listed in Table 2.

Table 2 Business requirements

Id	Requirement Description	Priority
B01	The system shall be able to collect and store system event logs	Must have
B02	The system shall be able to automatically analyze the system event logs to identify potential root causes of any issues or problems	Must have
B03	The system shall have models and algorithms	Must have
B04	The system shall be able to handle large volumes of system event logs and provide fast and reliable root cause analysis.	Must have
B05	The system should be easy to use, with a user-friendly interface that allows users to quickly and easily access the information they need	Nice to have

Functional Requirements

Functional requirements articulate the detailed functionalities and behaviors that the system should exhibit. Some of these requirements include:

F01: Anomaly Prediction Algorithm: The system's capability to predict anomalies by analyzing the frequency of logs within a specified timeframe.

F02: Log Parsing: The requirement for the system to efficiently parse logs, a foundational step for analysis.

F03: These requirements delineate the system's ability to identify different types of session vectors—normal, abnormal, and pollution data—contributing to precise analysis.

F04, F05, F06, F07: Visualization of event trees, linked events, and the option for filtering by process id or event id enhances user understanding and decision-making.

F08: Model Update: The ability to update the model based on user input enhances accuracy over time.

F09: Automated Reporting: The automatic generation of reports facilitates efficient communication and decision-making.

F10: Integration with ASML IT Systems: Check the possibility of integration with ASML's existing IT systems.

F11: Collection and Storage of System Event Logs: Collect and store system event logs from various sources.

F12: Automated Identification of Potential Root Causes: Automatically identify potential root causes from event logs.

F13: Use of AI/Mathematical Models for Root Cause Identification: Employ AI/mathematical models for root cause identification.

F14: Identification of Pollution Data: Identify pollution data within logs.

F15: Ensure high reliability and availability for uninterrupted access.

F16: Additional Reports for Developers

Description: Generate extra reports, including CPD execution phase, for developers.

Functional requirements are detailed statements of capabilities, behavior, and information that the solution should address. Table 3 lists the functional requirements of this project.

Table 3 Functional Requirements

Id	Requirement Description	Priority
F01	The algorithm shall predict the anomaly in logs by analyzing the number of logs in the time period	Must have
F02	The system shall parse the logs	Must have
F03	The system shall identify the abnormal event ids	Must have
F04	The system shall visualize the user about the event tree by process id	Could have
F05	The system shall visualize the linked events	Could have
F06	The visualizer shall be filtered by process id	Could have
F07	The visualizer shall be filtered by event id	Could have
F08	The system shall update the model if user input to result was false positive	Could have
F09	The system shall generate report about the results automatically	Must have
F10	The system should be compatible with existing ASML environment	Should have
F11	The system should be able to collect and store system event logs from various sources such as simulated test environment and cloud based simulated test environment results from testing management system	Must have
F12	The system shall have an automated analysis module that identifies potential root causes of any issues or problems from the collected system event logs.	Must have
F13	The analysis module shall use AI/ mathematical models and algorithms to identify the hint to root cause of issues or problems.	Must have
F14	The system shall identify pollution data	Must have
F15	The system shall be highly reliable and available to ensure that users can access it whenever they need it.	Should have
F16	The system should give some extra reports in addition to the hint root cause to developer to ease tracking the main root cause: like phase of the CPD sequence execution	Should have

Non-Functional Requirements

Non-functional requirements encompass quality attributes that underpin the system's overall performance, maintainability, and user experience. Some of these attributes include:

NF01, NF02: The imposition of execution time and memory limits safeguards optimal system performance.

NF03, NF04: The emphasis on maintainability and modularity guarantees the system's longevity and ease of management.

NF05: Clear Documentation: Thorough documentation ensures knowledge transfer and system sustainability.

NF06: Data Parsing and Quality: The system should accurately parse data and maintain data quality standards to ensure the reliability of processed information.

Table 4 lists the requirements that specify criteria used to judge the system's operation rather than specific functionality.

Table 4 Non-Functional Requirements

Id	Requirement Description	Priority
NF01	The system should not exceed a maximum execution time limit of 1 hour for any operation when running on a typical PC	Must have
NF02	The system should operate within a memory usage limit that does not exceed 1 megabyte (1 MB) when running on a typical PC	Must have
NF03	The system shall be designed to be easily maintained	Should have
NF04	The system shall be modular in design	Should have
NF05	The system shall have a clear and concise documentation	Must have
NF06	The system shall parse data correctly and ensure data quality	Should have

Architectural Requirement

Architectural requirements lay the foundation for a scalable, efficient, and effective system. Notable requirements include:

I01: Scalable Design: A design that accommodates future growth and changing demands ensures long-term relevance.

I02: Effective Data Management: The architectural approach should optimize data management for efficient operation.

Table 5 lists the architecturally significant requirements that should be taken into account while defining the system architecture.

Table 5 Architectural Requirements

Id	Requirement Description	Priority
I01	The system shall be designed to be scalable to accommodate future growth and changes in demand.	Must have
I02	The system's architectural design should include efficient data management capabilities, ensuring that data is stored, retrieved, and processed in a manner that minimizes latency, optimizes resource usage, and supports scalability.	Should have

Designing Feasible Solutions

Log analysis and root cause analysis have been active areas of research and development for many years. Various techniques and tools have been proposed and implemented to automatically analyze system logs and identify potential issues or anomalies. In this chapter, we discuss some of the related work and methods in the area of log analysis and root cause analysis, along with the feasibility of implementing such systems.

5.1 Feasibility Analysis

Many organizations already collect system logs and use various tools and techniques for log analysis and root cause analysis. However, designing and implementing a system that can handle high volumes of event data and provide fast and reliable analysis is a challenging task. The system must consider performance, scalability, and other design criteria/quality attributes.

One important factor to consider is the cost of implementing and maintaining the system. Depending on the size and complexity of the system, it may require significant resources in terms of hardware, software, and personnel. Organizations need to carefully evaluate the cost-benefit analysis of implementing such a system, taking into account factors such as the potential benefits in terms of improved system reliability and reduced downtime.

Moreover, the feasibility of the case depends on the specific requirements and constraints of the organization, as well as the availability of appropriate tools, technologies, and expertise. While there are many existing tools and systems available in the market that can be used for log analysis and root cause analysis, the choice of method depends on the specific requirements and constraints of the organization.
In conclusion, log analysis and root cause analysis are essential for maintaining and improving system reliability. While there are various techniques and tools available to perform such analysis, the choice of method depends on the specific requirements and constraints of the organization. Therefore, careful evaluation of the cost-benefit analysis is necessary before implementing such a system.

5.2 Challenges Arising from Real-World Requirements

In the domain of anomaly detection, a landscape intricately woven with technical nuances, significant challenges emerge as a direct result of operational imperatives within real-world settings. These challenges find their foundation in the insights uncovered in the preceding chapter. In our exploration of these challenges, we unearth pathways towards establishing resilient risk detection mechanisms coupled with strategic mitigation strategies. This subsection undertakes a comprehensive examination of the nuanced issues that manifest during the anomaly detection process. This examination is deeply informed by my prior experiential insights and remains closely aligned with the predefined technical requirements expounded upon in earlier sections. We explored some of the issues that arise when detecting anomalies and provide potential solutions.

Addressing False Alarms in the System:

The system changing can cause false alarms. To fix this, we update the model with new data and adjust its settings. Alternatively, we can use smarter models that adapt to changes without needing complete updates. This saves computing power and still catches anomalies accurately.

Mitigating Data Collection Costs:

Data collection can be expensive, particularly for large-scale systems. It is essential to determine the optimal balance between data quantity and quality. To mitigate this issue, we considered using a sampling approach to reduce data size, prioritize data collection based on critical system components.

Validating Performance Across Scenarios:

The model's performance across diverse scenarios is pivotal for anomaly detection. This challenge mandates formulating distinct test scenarios, utilizing metrics like accuracy.

Domain Knowledge and Feature Selection:

Accurate anomaly identification relies on understanding system behavior, particularly in complex setups. Appropriate feature selection is imperative to pinpoint critical components.

Table 6 Challenges in the Project

Challenge Description	**Category (Managemental/Technical)**	**Mitigation and Strategy**
Compatibility of Libraries and Dependencies	Technical	- Thoroughly assess library compatibility - Explore alternative library - Escalate compatibility issues
High Data Collection Costs	Technical	- Consider data sampling to reduce size - Prioritize critical data components - Seek cost-effective data sources
Admin Access Requirements	Technical	- Discuss admin access policies with the company - Seek workaround solutions if necessary
Lack of Diversity in the Dataset for Accurate Detection	Technical	- Ensure diverse and representative dataset - Data augmentation if necessary
Compatibility Issues with Software Versions	Technical	- Keep software versions updated - Test compatibility early on - Maintain documentation for versions
Implementation of Active Learning Methods	Technical	- Provide training and support for active learning - Explore alternative methods if necessary

Based on table 6, I listed the technical challenges encountered in implementing the system. Challenges include compatibility issues with libraries and dependencies, high data collection costs, administrative

access requirements, diversity in the dataset for accurate detection, compatibility issues with software versions, and effective implementation of active learning methods.

5.3 Risk Navigation

Risk management is an essential component of any project, and it involves identifying, assessing, and mitigating potential risks that may impact the project's success. In this thesis section, we discuss the risks and challenges associated with a cloud-based system aimed at detecting anomalies in a given dataset. We explore the various risks and challenges that may arise during the implementation of the project, including cloud-based system challenges, data gathering, licensing, and active learning.

One of the significant risks associated with a cloud-based system is the compatibility of libraries and dependencies. This risk is essential as the system may require various libraries and dependencies to function correctly, and any issues with compatibility may lead to errors, which can affect the project's success. Another significant challenge is the cost of data collection, which can be a significant barrier, particularly if the dataset is massive.

Another significant risk is the need to run some libraries and tools on company laptops that requires admin access. This requirement can create challenges, particularly if the company has strict policies regarding admin access. Data gathering is another challenge, as it requires a diverse and representative dataset to ensure accurate anomaly detection. The risk associated with data gathering is that if the dataset is not diverse, it may not be representative, leading to inaccurate anomaly detection.

The project's scope definition and budget are also essential factors to consider when assessing risks and challenges. A poorly defined project scope and budget can lead to poor project outcomes, particularly if the scope is too broad or the budget too small to cover the project's essential elements. Another significant risk is the need to retrain the model based on false positive tasks, which can increase the project's scope and make adding real-time capability to the system hard.

In conclusion, risk management is a critical component of any project, and it is particularly essential in cloud-based systems designed to detect anomalies in datasets. The risks and challenges associated with this type of project include compatibility of libraries and dependencies, data gathering, licensing, active learning, and versioning compatibility. To mitigate these risks, it is essential to assess alternative solutions, modify the project scope, and escalate issues that require attention from stakeholders. By addressing these risks and challenges, it is possible to ensure the project's success and achieve the project's objectives.

Table 7 Risks in the Project

Risk Description	**Category (Managemental/Technical)**	**Mitigation and Strategy**
Project Scope and Budget Issues	Managemental	- Define clear project scope and budget - Regularly review and adjust as needed - Involve stakeholders in scope decisions
Licensing Agreement Challenges	Managemental	- Ensure compliance with licensing agreements - Seek legal counsel if necessary

Based on table 7 I listed potential risks associated with the project, encompassing both managemental and technical domains. Risks include issues related to project scope and budget, licensing agreements,

and the need for frequent model retraining due to false positives. Addressing these risks is crucial to ensure the project's success and mitigate potential setbacks.

Risk Identification:

The VDI system's role in pivotal tasks, such as storing test results and ensuring compatibility, highlights its significance. Incorporating a cloud-based system introduces risks like compatibility issues, versioning complexities, licensing matters, and admin access needs.

Risk Mitigation and Strategy:

Risk mitigation strategies involve exploring alternatives, modifying project scopes, escalating issues, conducting thorough risk assessments, leveraging expertise and documentation, and meticulous project planning.

5.4 Prototype Blueprint

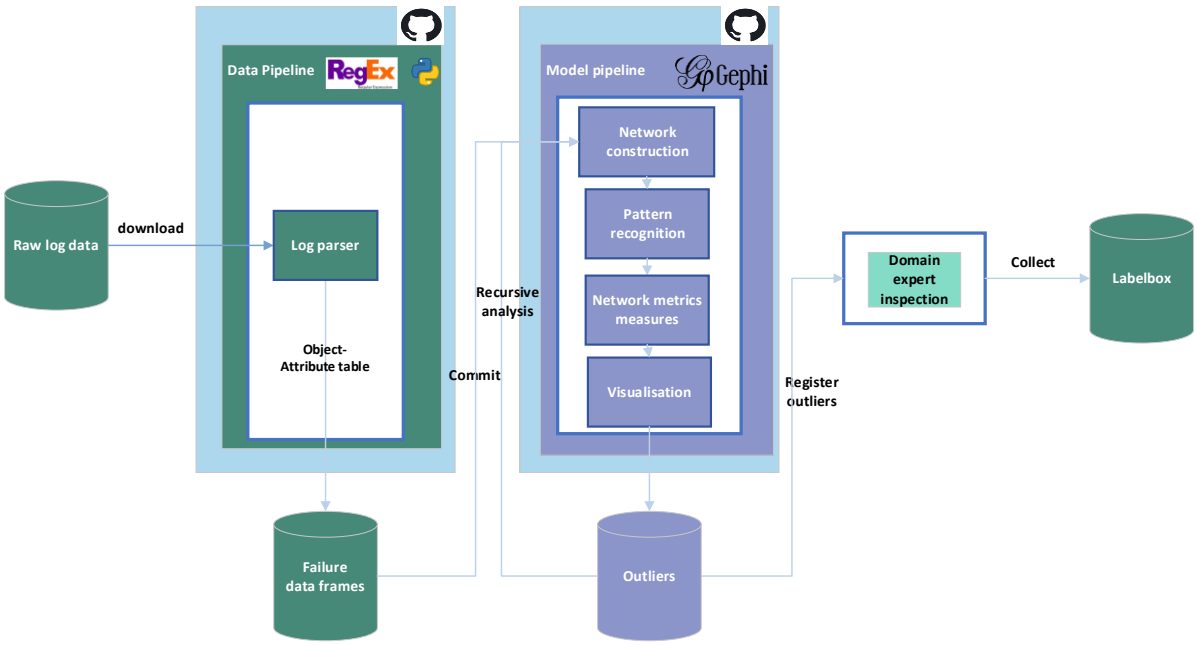

Figure 3 Prototype architecture

Figure 3 focuses on the design of a prototype aimed at mitigating risks in each phase of the automation of event log diagnosis project. The prototype is structured in a sequential manner, consisting of several key components.

Firstly, the prototype begins by taking input of raw log files, which serve as the initial data source for the analysis. These raw log files contain crucial information about system events and behaviors that require processing and interpretation.

Next, the prototype incorporates a data pipeline that handles the pre-processing, cleansing, and transformation of the log data. This pipeline ensures the data is properly formatted and prepared for subsequent analysis steps.

Following the data pipeline, a model pipeline is implemented, incorporating various analysis models and algorithms to extract meaningful insights from the processed log data. These models may include process mining, statistical analysis, or machine learning techniques, depending on the specific requirements and goals of the project.

To further enhance the accuracy and reliability of the results, the prototype incorporates a critical step of domain expert inspection. This involves domain experts who possess in-depth knowledge and understanding of the ASML system and event log data. Their expertise and insights play a vital role in validating the analysis results and detecting any potential inaccuracies or anomalies.

Lastly, the prototype includes the design of a label box, which facilitates the annotation and labeling of the analysis outcomes. This labeling process enables the categorization and identification of specific events, patterns, or anomalies discovered in the log data.

By implementing this prototype design, the project aims to minimize risks at each phase, ensuring that the analysis results are accurate, reliable, and aligned with the requirements of the automation of event log diagnosis.

Data Pipeline

The data pipeline involves the collection and processing of test result data. The following steps will be taken to mitigate risks in the data pipeline:

- Data Collection: We collected data from a diverse and representative dataset that is relevant to the ASML setup sequence. The data was collected from multiple sources to ensure its completeness and accuracy.

- Data Structure: The data is structured and cleaned to ensure its quality and consistency. That was divided into test results and event logs.

- Log Parser: We used a log parser tool to extract meaningful information from the event logs. This tool helped us to identify the patterns and trends in the data.

Model Pipeline

The model pipeline involves the construction of a network that can identify abnormal behavior and diagnose faults in the system. The following steps will be taken to mitigate risks in the model pipeline:

- Network Construction: We will construct a network using the extracted information from the event logs. The network will be a directed graph with nodes representing the different components of the ASML system and edges representing the causal relationships between them.

- Pattern Recognition: We will use machine learning techniques to identify patterns and trends in the data. We will also use anomaly detection algorithms to identify abnormal behavior in the system.

- Visualization Analysis: We will use visualization techniques to analyze the network and identify the critical components of the ASML system.

- Network Measure: We will use network measures such as modularity class, clustering coefficient, and eccentricity to quantify the properties of the network.

- Domain Experts Inspection: We will seek the input of domain experts to validate the results of the diagnosis and provide additional insights into the system.

5.5 Conclusion

In conclusion, the intelligent diagnosis of event logs in ASML required a systematic and structured approach that involves the careful management of the data and model pipelines. By following a rigorous methodology, we identified risks and potential issues in the system and develop effective strategies to mitigate them. The success of the diagnosis relies on the accuracy and reliability of the data and the effectiveness of the model in identifying abnormal behavior and diagnosing faults in the system.

Design and Implementation

The design and implementation of the intelligent diagnosis tool play a pivotal role in enabling effective analysis of network data. This chapter outlines the architecture, components, and processes involved in the development of the Data Pipeline and Model Pipeline. The chapter also highlights the quality attributes guiding the design process.

6.1 Intelligent Diagnosis Tool

The intelligent diagnosis tool is a comprehensive solution for analyzing network data, comprising the Data Pipeline and Model Pipeline. It enables users to gain valuable insights into the network structure through efficient data processing, graph creation, and analysis techniques.

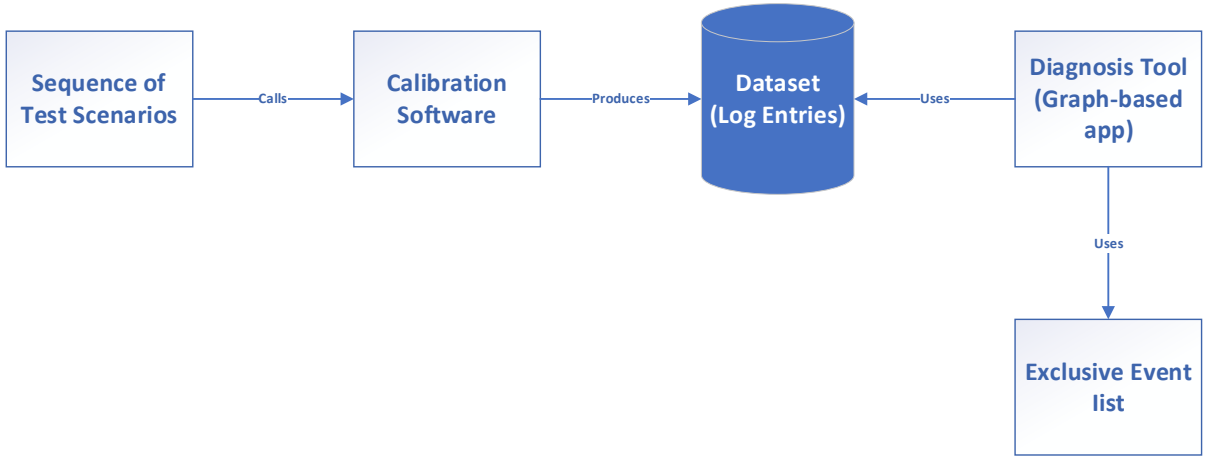

Figure 6 Intelligent diagnosis tool integration diagram

6.2 Implementation Phases

Before delving into the specific phases of implementation, it's crucial to understand why we chose the methodology underpinning this intelligent diagnosis tool. The selection of this methodology was driven by the unique structure of the ASML data, which involves events intricately linked together. These links are explicitly defined within the process.

Why Graph Theory-Based Solutions?
The ASML data structure exhibits a network-like quality where certain events are interconnected, akin to nodes in a graph. The relationships between these events are vital and provide invaluable insights for analysis. Leveraging graph theory-based solutions, we can effectively model this interconnectedness as edges in a graph.

Advantages:

- Efficiency and Speed: Representing events as nodes and their connections as edges allows for efficient calculations and reduced computational complexity. By assigning weights to edges based on the significance of the connections, we can optimize the analysis process.

- Simplicity and Lower Computational Complexity: Graph theory simplifies the data representation, leading to lower computational complexity compared to training-intensive machine learning methods. The method shines in capturing the essential relationships without unnecessary intricacies.

This approach essentially transforms the data into a graph representation where events are nodes and links between them are edges. By prioritizing important nodes with lower weights, we can achieve efficient and rapid results compared to the training requirements of machine learning-based approaches.

With this understanding, let's proceed to explore the specific phases of implementation.
The development of the intelligent diagnosis tool involved several key phases, each focusing on specific aspects of the system's functionality. These phases include:

a. Data Processing Phase

The Data Processing Phase is a fundamental step in our analysis pipeline, representing the initial processing of raw input data. This phase is essential to transform the raw data into a structured and organized format that is suitable for subsequent analysis. The primary objectives of this phase are to preprocess the data, parse it based on a specific format, construct the base network data, and potentially create a configurable network based on time considerations.

1. **Preprocessing the Data**

   We start by preparing the raw data for analysis. This involves cleaning, standardizing, and addressing any missing or inconsistent values. Essentially, we ensure the data is in good shape for further steps.

2. **Parsing the Data According to a Specific Format**

   We interpret the data based on a predefined structure. For our context, this structure is the Qualification Queue Test Event Log format in ASML. Parsing helps us extract meaningful information and organize it appropriately.

3. **Constructing the Base Network Data Using Graph Theory Principles and Assigning Edge Weights**

   Here, we use principles from graph theory to build the foundational network structure. Nodes represent specific data elements, and edges depict relationships between them. Assigning weights to edges helps us emphasize the importance of these relationships for a more insightful analysis.

4. **Creating a Customizable Network Based on Time**

   We consider time as a key factor. By organizing the data to account for time, we can analyze how events unfold over different time intervals. This time-based approach adds a valuable dimension to our analysis.

In summary, this phase transforms raw data into an organized, structured format, making it ready for the subsequent analysis stages. Each step contributes to understanding the data better and extracting meaningful insights from it.

b. Network Data Construction Phase

During the Network Data Construction Phase, we actively construct the foundational network data structure. This phase plays a pivotal role in enabling effective network analysis by building nodes and edges that contain crucial information.

1. **Creating Nodes and Edges**

   We actively generate nodes to represent individual elements or events from the processed data. These nodes serve as fundamental building blocks, encapsulating specific information. Additionally, we create edges to depict the relationships between these events. This process allows us to visually and conceptually represent how events are interconnected.

2. **Adding Attributes for Relationships and Metrics**

   To enhance the depth of our analysis, we attach attributes to the nodes and edges. These attributes provide additional context, such as the nature or strength of the relationship between

events. We also incorporate metrics, such as frequency or importance, enriching the network and enabling a more comprehensive analysis.

The Network Data Construction Phase is crucial as it actively transforms processed data into a meaningful network representation. It's akin to assembling a puzzle, where each node and edge contributes to the larger picture, allowing us to derive insights and detect patterns within the data.

c. Graph Creation Phase

In the Graph Creation Phase, we focus on converting the constructed network data into a usable graph format, ready for analysis. This phase is essential because it allows us to work with the data in a structured and flexible manner.

1. Abstracting and Configuring Graph Types

The GraphFactory class actively abstracts the creation of various graph types based on the network data. Different types of graphs can highlight different aspects of the data, enabling a more tailored analysis. By abstracting this process, we ensure the tool's adaptability and versatility, capable of handling various network representations.

2. Loading Network Data and Facilitating Subsequent Analysis

The GraphLoader class actively loads the constructed network data into a graph object. This step is vital as it prepares the data for in-depth analysis. Once the network is in a graph format, we can easily perform a range of analyses, from calculating centrality to identifying communities within the network.

The Graph Creation Phase is crucial in actively ensuring that the data is presented in a format that can be readily explored and analyzed. By providing options for different graph types and efficiently loading the data, this phase enhances the tool's capabilities and facilitates a more insightful analysis.

6.3 Data Pipeline

In this section, the data pipeline's implementation phases are detailed, encompassing two main modules: Data Processor and Base Network Data. The Data Processor module focuses on parsing and preprocessing input data, while the Base Network Data module is responsible for extracting relevant information, building the network structure, and incorporating additional attributes for meaningful analysis.

- **Data Processor**
  1. Parsing the Input Data
     The Data Processor class reads and interprets input files of various formats, extracting relevant fields for further processing.

  2. Preprocessing the Data
     Data preprocessing includes cleaning and standardizing data, handling missing values, and preparing it for transformation.

  3. Data Transformation
     The transformed data is organized into a structured format suitable for network analysis, featuring relevant attributes and dimensions.

  4. Data Output
     Methods for retrieving processed data or writing it to output files are provided for seamless integration.

- **Base Network Data**
  1. Extracting Relevant Information
     The BaseNetworkData class selects pertinent fields from processed data, such as event IDs, linked events, and timestamps.

  2. Building Network Data
     Nodes and edges are constructed based on the extracted information, forming the foundational structure of the network.

  3. Adding Attributes
     Additional attributes are incorporated into nodes and edges to capture relationships and provide context for analysis.
  4. Data Output
     Methods for retrieving built network data or exporting it for further use are implemented.

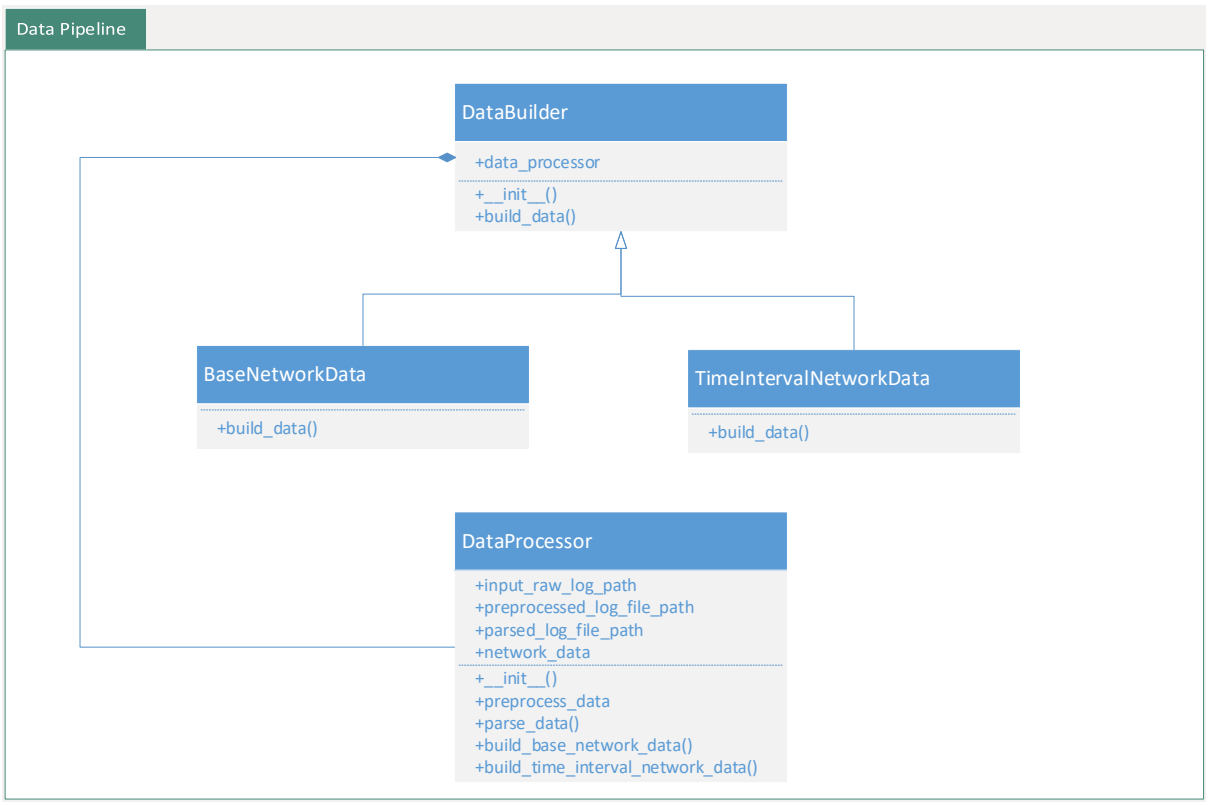

Figure 7 Data Pipeline class diagram

6.4 Model Pipeline

This section introduces two critical modules: Graph Factory and Graph Loader. The Graph Factory module facilitates the creation of diverse graph types, offering configurability for edge weights and visualization settings. On the other hand, the Graph Loader module focuses on converting base network data into a graph structure, enabling analysis such as centrality calculations, community identification, and graph visualization for deeper exploration of network properties.

- Graph Factory
  1. Graph Creation
     The GraphFactory class supports the creation of various graph types, abstracting implementation details and accommodating configuration parameters.

2. Configuration Management
   Flexible configuration options enable customization of graph creation, edge weights, and visualization settings.

- Graph Loader
  1. Loading Network Data
     The GraphLoader class translates base network data into a graph structure, encompassing nodes, edges, and relationships.

  2. Graph Analysis
     Methods for calculating centrality, identifying communities, and visualizing the graph aid in exploring network properties.

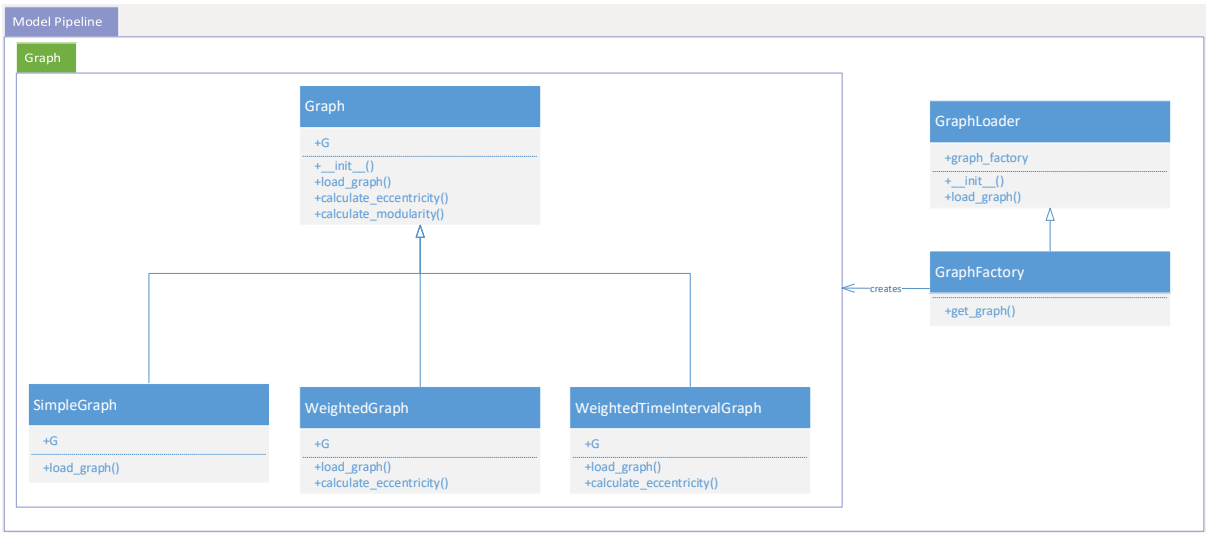

Figure 8 Model Pipeline class diagram

6.5 Data Processing and Anomaly Detection Workflow

This section shows runtime interactions among classes within a software system, aiming to provide a detailed understanding of the system's behavior under varying use cases. The point is the runtime interaction diagram, a dynamic representation illustrating the communication and collaborations between different classes during the execution of specific functionalities. By examining these interactions, we gain valuable insights into how the system functions and how various components work together to fulfill distinct requirements.

In this section, we present the diverse use cases that have been examined to analyze the runtime interactions of different classes within the system. Each use case represents a specific scenario or functionality, showcasing distinct class collaborations and highlighting their roles in fulfilling the respective requirements.

- **Use Case 1: Data Processing and Analysis**

  This use case revolves around the processing and analysis of raw data to derive meaningful insights. The runtime interaction diagram illustrates how classes related to data processing collaborate to transform raw data into valuable information for further analysis.

- **Use Case 2: Network Construction**

The network construction use case focuses on building a network representation from processed data. The runtime interaction diagram sheds light on the interactions between classes responsible for constructing the network and managing its components.

- **Use Case 3: Anomaly Detection**

  Anomaly detection involves identifying irregular patterns within the data. The runtime interaction diagram elucidates how classes collaborate to detect anomalies, showcasing the flow of information and decision-making processes.

Runtime Interaction Diagram

This section presents the runtime interaction diagram, providing a visual representation of class interactions for the aforementioned use cases. Each diagram is analyzed in detail to explain the interactions and their significance in fulfilling the respective use case requirements.

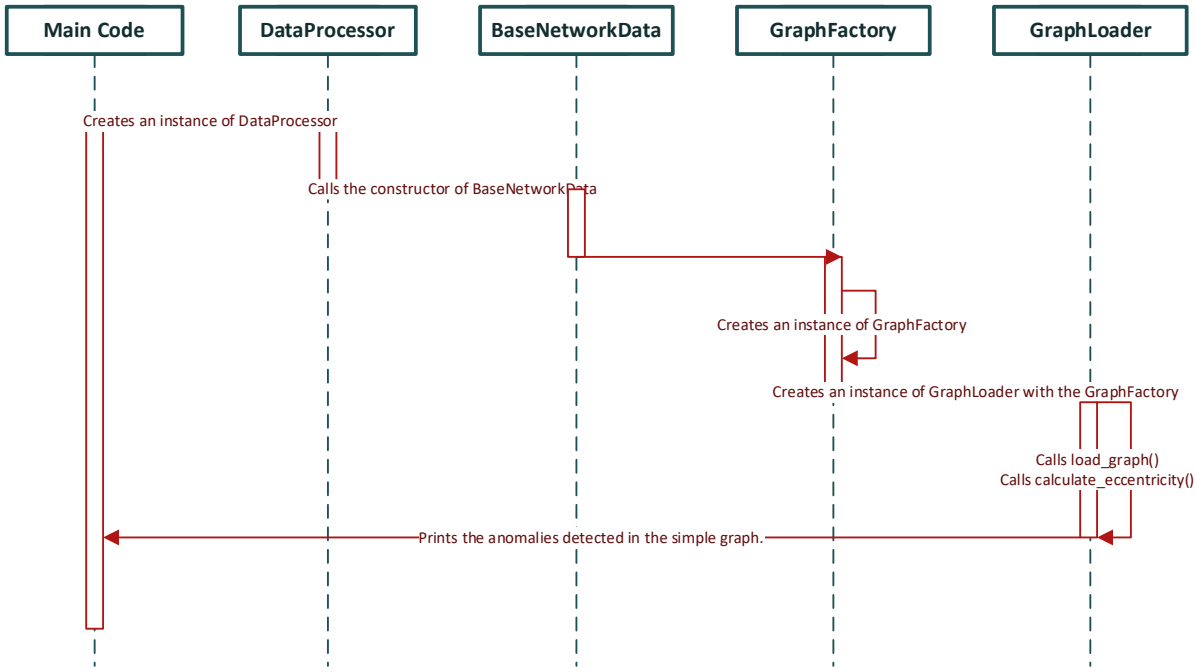

Figure 4 Runtime UML Sequence Diagram

Based on Figure 4, in the main section of the code, executed when the script is run directly, an instance of DataProcessor is created with designated file paths for input event logs, preprocessed logs, parsed data frames, and network data. The process_data method within DataProcessor is subsequently invoked. This method involves handling the provided file paths, including the initialization of BaseNetworkData by calling its constructor with the processed data. The build_data method within BaseNetworkData is then called, facilitating the construction of the network data.

This interaction demonstrates the pivotal role of DataProcessor in orchestrating data processing and its seamless integration with BaseNetworkData, enabling efficient network data construction. Additionally, GraphFactory is instantiated to support the creation of graphs, and GraphLoader is established using GraphFactory. During the loading of graphs using the load_graph method in GraphLoader, network_data and a specified graph type are provided as parameters. The

calculate_eccentricity method of the loaded graph is subsequently invoked, illustrating the step-by-step graph loading and analysis process. Ultimately, in the main section, anomalies detected in the simple graph are printed, showcasing the successful execution of the anomaly detection process in this scenario.

The sequence diagram depicts the primary interactions and data flow within the main file of the provided Python code. It begins with the initialization of the data processing using DataProcessor, encompassing the handling of event logs and data preparation. Following this, BaseNetworkData constructs the network data based on the processed information. The diagram then showcases the steps involving the GraphFactory and GraphLoader, which load network data and analyze it to detect anomalies within a simple graph.

6.6 Summary

This chapter provided an in-depth exploration of the design and implementation of the Data Pipeline and Model Pipeline within the intelligent diagnosis tool. The Data Pipeline encompasses data processing, transformation, and network data construction, while the Model Pipeline focuses on graph creation, loading, analysis, and visualization. The tool's design adheres to quality attributes such as scalability, modularity, flexibility, performance, usability, and extensibility. The next chapter evaluates the tool's effectiveness through experimental analysis and real-world case studies.

Verification and Validation

The thorough verification and validation of the graph-based intelligent diagnosis tool involved a meticulously designed set of test cases. These test cases were carefully crafted to assess the tool's capability to fulfill the specified functional and non-functional requirements. By subjecting the tool to diverse scenarios and data inputs, we aimed to ensure its accuracy, reliability, and adherence to performance constraints.

7.1 Datasets Description

My investigation revolves around three distinct datasets originating from unique database identifiers. These datasets capture a wide array of event types and occurrences. The count of events within each dataset, coupled with the labeled root cause event IDs derived from domain knowledge experts' analysis. Guided by domain knowledge experts' analysis of realistic queue qualification tests and similar scenarios, we meticulously crafted a series of test cases. These test cases were curated to simulate a spectrum of potential scenarios, accounting for both common and exceptional cases. This approach ensured that our analysis would capture a holistic understanding of the datasets' nuances and behaviors, consistent with established industry standards.

7.2 Test Cases and Verification and Validation

7.2.1 Anomaly Detection (F-01)

The analysis of event logs plays a pivotal role in modern industries, enabling the extraction of valuable insights from the vast amounts of data generated by intricate processes. This section focuses on the verification and validation of the intelligent diagnosis tool's anomaly detection capabilities within the context of event log analysis at ASML. ASML, a leading semiconductor equipment manufacturer, operates highly complex and interconnected systems where anomaly detection is crucial to ensure optimal performance, mitigate risks, and maintain manufacturing precision.

7.2.1.1 Test Case: VV-AD-01 - Scenarios Variation

Description: ASML's operational environment involves diverse scenarios with varying numbers of events, each contributing to the intricate web of operations. The ability of the intelligent diagnosis tool to accurately detect anomalies under these dynamic conditions is of utmost importance to enhance production efficiency and minimize disruptions.

Methodology: A range of scenarios was meticulously designed, simulating different operational contexts at ASML. These scenarios encompassed varying event counts, from small-scale to large-scale inputs, and integrated both abnormal and normal events.

Expected Outcome: The tool demonstrated its capability to effectively identify anomalies across a spectrum of scenarios, showcasing its adaptability and robustness in dealing with the intricacies of ASML's event logs.

Table 6 Key Dataset Details

Dataset id	Total count of events	Labeled root cause coded event id
1	2916	E136 E000
2	22230	E0100
3	4308	E0608 E0310 E0000 E0082

4	67658	Can’t infer anything from the ER events "compare queue results" failed
5	64354	Can’t infer anything from the ER events "compare queue results" failed
6	2995	E0004 E847 E840 E0605 E00000

7.2.1.2 Test Case: VV-AD-02 - ER Event Log Analysis
Description: ASML's event logs contain a rich tapestry of operational data, reflecting the interactions and dependencies within its manufacturing processes. This test case scrutinized the tool's ability to accurately detect anomalies within ASML's complex and authentic ER event log datasets.

Methodology: A diverse set of ER event log datasets, representative of different ASML operations, were employed for analysis. The tool's anomaly detection performance was evaluated in terms of accuracy and consistency across these datasets.

Expected Outcome: The tool consistently and accurately identified anomalies within ASML's ER event log datasets, affirming its effectiveness in deciphering complex operational patterns.

Table 7 Intelligent diagnosis tool results based on different scenarios

Task ids	**1**	**2**		**3**		**4**	**5**	**6**
Known issue	E0608 E0310 E0000 E0082	Can’t infer anything from the ER events "compare queue results" failed		Can’t infer anything from the ER events "compare queue results" failed		E136	E0100	E0004 E847 E840 E0605 E00000
Eccentricity app result	('E0608' 120) ('E0310' 100) ('E0000' 80) ('E0082' 60)	**Based on overall ER event log results:** ('E0100' 180) ('E0401' 160) ('E0400' 140)	**Failed subtest result:** **threshold is less than 20 => no detected event error in test results**	**Based on overall ER event log results:** ('E0100' 140) ('E009' 120) ('E0401' 120)	**Failed subtest result:** **threshold is less than 20 => no detected event error in test results**	E136	E0100	E0004 E847 E840 E0605

7.2.2 Pollution Event Identification (F-06) - Ensuring Precision in ASML Operations
An accurate assessment of pollution events is of paramount importance within ASML's manufacturing processes, where even minor deviations can have significant implications. This section focuses on the validation of the intelligent diagnosis tool's ability to precisely identify pollution events within ASML's event logs.

7.2.2.1 Test Case: VV-PEI-01 - Pollution Data Verification
Description: The analysis focuses on the tool's ability to accurately detect pollution events within event log data, an essential aspect for ASML's operations, necessitating swift and precise identification.

Methodology: Event logs were tailored to include instances of pollution events, replicating various pollution scenarios that may appear in the event logs relevant to ASML's operations. The tool's ability to accurately pinpoint these pollution events was meticulously evaluated.

Expected Outcome: The tool adeptly identified pollution events within ASML's event logs, attesting to its precision in detecting and mitigating environmental anomalies.

Table 8 Pollution event logs data and accuracy of events detection

Pollution events list	**Total number detected events ids(repetitive)**	**Total number detected events ids(unique)**	**Accuracy of subset of 10**	**Accuracy of subset of 15**	**Accuracy of subset of 20**
	659	180	100	100	100

7.2.3 Automatic Report Generation (F-07)
The ability to generate accurate and consistent reports is crucial for facilitating informed decision-making. To verify this capability, the following test case was conducted:

Test Case: VV-ARG-01 - Report Generation Consistency
Description: This test case focused on evaluating the system's report generation capabilities using different event logs, with an emphasis on the consistency of the generated reports.

Methodology: We subjected the tool to various event logs, each representing distinct network scenarios. The generated reports were compared to assess their coherence and alignment with the analyzed data. Check the file format, execute different tests, cross-reference with original file, expected results, timeline and error handling are the actions we took in tool design to guarantee the report generation consistency.

Expected Outcome: The generated reports consistently reflected the analyzed data, demonstrating the tool's proficiency in automatic report generation.

7.2.4 Performance and Memory Usage (NF-02, NF-03)
The performance and memory usage of the intelligent diagnosis tool are pivotal factors in its overall utility. To gauge these aspects, the following test cases were executed:

Test Case: VV-PMU-01 - Response Time

Evaluation
Description: This test case involved the evaluation of the system's response time to ensure it met the specified execution time limit.

Methodology: The tool was subjected to various scenarios, each representing different operational contexts. The response time for each scenario was measured and compared against the predetermined execution time limit.

Expected Outcome: The system's response time was found to be within the specified execution time limit, meeting performance expectations.

Test Case: VV-PMU-02 - Memory Usage Monitoring
Description: This test case aimed to monitor and measure the tool's memory usage during different phases of execution.

Methodology: The system's memory usage was tracked using appropriate monitoring tools and techniques. Memory measurements were taken during various stages of execution to ensure compliance with memory usage constraints.

Expected Outcome: The tool's memory usage remained within acceptable limits throughout execution, confirming its efficient resource management.

Table 8 Response time and memory usage based on each scenario

Test Scenario	**Response time(in seconds)**	**Memory usage(KB)**
1	0.66602	116.05859375
2	6.44831	117.546875
3	0.58472	116.50390625
4	12.46434	118.30859375
5	11.22432	118.2578125
6	0.25909	114.96484375

Based on table 8, I utilized the memory_profiler.memory_usage() function from the memory_profiler Python library to gauge the memory consumption of my Python program. The function allows real-time monitoring of memory usage during program execution. I strategically placed memory measurement points within the codebase, particularly at critical stages of computation. These designated locations were marked using decorators, specifically @profile, enabling the tool to capture memory snapshots at those instances. It's important to note that these measurements were conducted on my personal PC during regular usage and not during peak server loads. Despite this, the recorded memory usage values, represented in megabytes, offered valuable insights into the program's memory consumption patterns. This data was instrumental in identifying memory-intensive segments and potential optimization areas, even within the context of personal computer usage.

7.2.5 Data Parsing Accuracy (NF-08)
Accurate data parsing is fundamental for extracting meaningful insights from network data. To validate this core functionality, the following test case was conducted:

Test Case: VV-DPA-01 - Data Parsing Precision
Description: This test case involved the provision of diverse datasets with different formats and structures to verify the tool's accurate parsing of data.

Methodology: We curated datasets encompassing varying data formats and structures, representing real-world data heterogeneity. The tool's parsing results were meticulously compared against the original datasets.

Expected Outcome: The tool exhibited precise data parsing capabilities, accurately handling diverse data formats and structures.

Table 9 accuracy of tool's parsing module

Test Scenario	Total count of events	Total number of correctly parsed events	Parsing accuracy rate
1	2916	2719	93.24%
2	22230	21796	98.05%
3	4308	4145	96.21%
4	67658	66256	97.92%
5	64354	62888	97.72%
6	2995	1774	66.7%

In scenario 6, as per Table 9, log parsing accuracy is comparatively lower due to the distinct log format and structure inherent to this specific scenario. The discrepancy arises from its differing log format compared to the focus of this project, which centers on realistic queue tests, as opposed to the alternative test type represented by scenario 6.

7.3 Summary

This chapter presented a comprehensive verification and validation process conducted on the intelligent diagnosis tool. The devised test cases encompassed a wide array of functional and non-functional requirements, rigorously assessing the tool's performance, accuracy, and adherence to constraints. The outcomes of these test cases provide compelling evidence of the tool's reliability, effectiveness, and capacity to empower network diagnostics through advanced graph-based analysis.

In the subsequent chapter, the results of experimentation and analysis will be detailed, shedding further light on the tool's practical performance and real-world applicability.

The verification and validation of the intelligent diagnosis tool's capabilities, particularly in anomaly detection and pollution event identification, yielded promising results within the context of event log analysis at ASML. Through a systematic approach, a comprehensive set of test cases was executed, rigorously evaluating the tool's performance, accuracy, and adherence to constraints.

In the realm of ASML's operations, where precision and efficiency are paramount, the tool demonstrated its potential to enhance anomaly detection and pollution event identification. These findings underscore the tool's relevance in real-world industrial applications and lay the foundation for its effective integration into ASML's operational framework.

The next chapter explores the results of the experiments and analyses, providing a closer look at how the tool performs in practice and its potential impact on ASML's operations.

Conclusion

The utilization of advanced data analysis techniques within industrial operations has become a cornerstone for enhancing efficiency, reliability, and overall performance. In the context of ASML's intricate manufacturing processes, the application of event log analysis has been explored through the lens of a graph theory-based intelligent diagnostic tool. This chapter provides a comprehensive conclusion, encapsulating the outcomes, insights, and implications derived from the study.

8.1 Results

The rigorous validation and experimentation of the graph theory-based intelligent diagnostic tool within the realm of ASML's Realistic Queue Qualification Tests have yielded noteworthy results and significant observations. The tool's anomaly detection capabilities demonstrated remarkable accuracy across diverse scenarios, underscoring its potential to enhance early detection of irregularities in complex manufacturing processes. The precise identification of pollution events within ASML's event logs signifies a critical step towards proactive risk management. Furthermore, the automatic report generation feature showcased the tool's ability to synthesize complex data analyses into actionable insights, expediting decision-making processes.

The evaluation of the tool's performance and memory usage aligns with the stringent demands of ASML's operations. Its efficient resource utilization and compliance with specified execution time limits attest to its suitability for real-time applications within industrial contexts.

8.2 Conclusion

In conclusion, the integration of the graph theory-based intelligent diagnostic tool into ASML's manufacturing operations has proven to be a practical and valuable addition. The tool's notable contributions include a significant reduction in investigation time, facilitated by its ability to efficiently detect anomalies and consider their weights to discern event patterns and potential failure scenarios.

The tool's foundation in graph theory has enabled a clear and insightful visualization of event log data, aiding in the understanding of event interconnections. Its lightweight design, devoid of the need for extensive training phases, ensures adaptability in handling evolving log sequences, aligning well with the changing manufacturing environment.

A key strength lies in the tool's real-time functionality, allowing seamless integration with various behaviors and scenarios. This adaptability is crucial for effectively addressing the dynamic nature of the manufacturing environment. The successful application of this tool within ASML's Realistic Queue Qualification Tests underscores its potential to streamline manufacturing processes, reduce risks, and foster a culture of continuous improvement. As ASML continues to push the boundaries of innovation, the graph theory-based intelligent diagnostic tool represents an effective convergence of data analysis and industrial operations, paving the way for operational efficiency and informed decision-making.

8.3 Recommendation and future work

While the graph theory-based intelligent diagnostic tool has demonstrated promising capabilities, there exist avenues for further enhancement and exploration:

- **Enhanced Event Visualization:**
  Future development should prioritize enhancing event visualization within the tool using graph theory. Incorporating a feature that clearly represents event relationships and links will provide stakeholders with a more intuitive understanding of the data. This enhancement becomes even more impactful when accessible through a user-friendly web application, enabling a broader audience to benefit from the tool.

- **Cross-Platform Compatibility:**
  To ensure broader accessibility and seamless integration, efforts should be made to create a self-contained executable version of the tool compatible with both Windows and Linux platforms. Cross-platform compatibility will enhance the tool's versatility, making it adaptable to various testing environments and catering to a wider user base.

- **Exploration of Additional Graph Metrics:**
  Future research should delve into exploring and utilizing a diverse set of graph theory metrics beyond eccentricity. This exploration will enable a more comprehensive analysis of different scenarios, providing a deeper understanding of event interdependencies. Reporting on and analyzing these additional metrics will enhance the tool's analytical capabilities and the insights it offers.

- **Collaborative Knowledge Sharing and Integration:**
  Encouraging collaboration among multiple ASML teams involved in diagnostics is crucial. Establishing a centralized knowledge repository or a collaborative platform where teams can share insights, labeled datasets, and root causes based on various scenarios is recommended. Additionally, integrating machine learning techniques for failure prediction and root cause detection, leveraging the collective expertise, can further enhance the tool's capabilities and its potential for driving operational excellence.

In summary, implementing the outlined recommendations will significantly enhance the tool's usability, analytical depth, and collaborative potential. These steps aim to refine event visualization, achieve cross-platform compatibility, explore diverse graph metrics, and promote knowledge sharing. By embracing these improvements, ASML can harness the tool's full potential to drive operational efficiency and informed decision-making in manufacturing processes.

Abbreviations and Glossary

Test cases naming and abbreviation
The choice of test case names was based on a systematic and descriptive approach that aimed to clearly convey the purpose and scope of each test case. Here's how the test case names were derived and why they were chosen:

Test Case Prefixes:
The test case names start with a prefix that indicates the type of test case. For example, "VV" stands for "Verification and Validation," which provides context that these are test cases related to the validation process.

Functional Requirement Codes:
The functional requirement codes (e.g., F-01, F-06) were incorporated into the test case names to directly link the test case to the specific functional requirement being tested. This helps in clearly identifying which aspect of the system's functionality is being verified.

Brief Description:
Following the requirement code, each test case name includes a brief description that summarizes the objective of the test case. This description is concise yet informative, giving a clear idea of what the test case aims to achieve.

Context or Focus:
In some cases, the test case name includes additional information about the context or focus of the test. For instance, "Scenarios Variation" in "VV-AD-01" highlights that the test is focused on different scenarios, and "ER Event Log Analysis" in "VV-AD-02" specifies the type of event logs being analyzed.
The chosen approach ensures that the test case names are meaningful, informative, and easy to understand. They provide a quick snapshot of the test's purpose, the aspect being tested, and any specific context. This aids in clear documentation, communication, and understanding of the verification and validation process.

Glossary

CPD	Calibration, Performance and Diagnostic application
IC	Integrated Circuits
ER	Error Recording
GNN	Graph Neural Network
PSG	Project Steering Group
PCA	Principal Component Analysis
LSTM	Long Short-Term Memory
EUV	Extreme Ultraviolet
FC	Functional Cluster
SNA	Social Network Analysis
SW	Software
VDI	Virtual Desktop Infrastructure
CC	Clear Case
TM	Test Management
TH	Trace Handler
MoSCoW	M – Must have, S – Should have, C – Could have, W - Won't have

References

[1] Annual Report, ASML. "Small patterns. Big impact. ." 2022.
[2] Jeremy Dick, Elizabeth Hull , Ken Jackson. *Requirements Engineering*. Springer, 2017.
[3] Zhuangbin Chen, Jinyang Liu, Wenwei Gu, Yuxin Su, Michael R. Lyu. "Experience Report: Deep Learning-based System Log Analysis for Anomaly Detection." 2021.
[4] Zhao, Z., Xu, C., & Li, B. (2021). A LSTM-based anomaly detection model for log analysis. Journal of Signal Processing Systems, 93, 745-751.
[5] Cinque, Marcello, Domenico Cotroneo, and Antonio Pecchia. "Event logs for the analysis of software failures: A rule-based approach." IEEE Transactions on Software Engineering 39.6 (2012): 806-821.
[6] Soboleva, A., & Tushkanova, O. (2020, April). The Methodology of Extraction and Analysis of Event Log Social Graph. In 2020 26th Conference of Open Innovations Association (FRUCT) (pp. 415-422). IEEE.
[7] Yuan, Y., Wang, Z., & Wang, Y. (2022). Learning latent interactions for event classification via graph neural networks and PMU data. IEEE Transactions on Power Systems, 38(1), 617-629.
[8] Yadav, R. B., Kumar, P. S., & Dhavale, S. V. (2020, June). A survey on log anomaly detection using deep learning. In 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO) (pp. 1215-1220). IEEE.
[9] Min, S., Gao, Z., Peng, J., Wang, L., Qin, K., & Fang, B. (2021). STGSN—A Spatial–Temporal Graph Neural Network framework for time-evolving social networks. Knowledge-Based Systems, 214, 106746.

Appendix A. Stakeholder Analysis

Stakeholder analysis is an essential step in any project as it helps to identify and understand the needs and expectations of individuals or groups that can affect or be affected by the project. In this report, we present a detailed stakeholder analysis of the project, which involves ASML Netherlands B.V. and the Eindhoven University of Technology. This chapter aims to provide an in-depth understanding of the stakeholders' interests, concerns, and roles in the project. It also outlines the process of stakeholder identification, analysis, and engagement. The main stakeholders in this project include representatives from different departments of ASML, such as the software development team, the quality assurance team, and the engineering team. The Eindhoven University of Technology is also a significant stakeholder, as it provides research support and academic expertise. The stakeholder analysis chapter provides a comprehensive overview of the stakeholders' roles, responsibilities, and expectations, which helped to ensure that the project meets all the relevant requirements and expectations.

8.4 ASML Netherlands

ASML Netherlands B.V. is the industrial part of this project. The outcome of this project could bring added value to the company. ASML stakeholders were responsible for providing the details and related domain knowledge of the root cause analysis and realistic queues. Table 6 lists the main ASML stakeholders. Their involvement level was quite different depending on their expertise and interest. Mari Mnatsakanyan, Andre Korbes, and Junchao Xu, the project supervisors, closely monitored the project's progress via weekly meetings. The rest of the stakeholders were involved in the project through necessary meetings and project steering group meetings.

Table 8 ASML stakeholders

Id	Role	Concerns
1	ASML Supervisor	A. Ensuring the system design satisfies stakeholders' concerns B. Helping the EngD trainee to overcome a steep learning curve inside ASML
2	ASML Supervisor	A. Ensuring the system design satisfies stakeholders' concerns B. Helping the EngD trainee to overcome a steep learning curve inside ASML
3	ASML Supervisor	A. Ensuring the system design satisfies stakeholders' concerns B. Helping the EngD trainee to overcome a steep learning curve inside ASML
4	ASML Group Leader	A. Project success B. Adherence to the company rules
5	ASML Software Field Performance - Analytics and Visualization Engineer in SQL Test Data Analysis team	A. Visualization effect B. Manual inspection importance

6	ASML Engineer - Automatic root cause analysis Engineer	A. Graph theory based method Verification B. Detecting pollutions C. Helps in comparative analysis between different automation tools
7	Software development, requirement analysis, implementation and design	A. Software development technical limitation B. Comparative analysis of different root cause automation tools C. Verification methods
8	Data science engineer	A. Casual influence context B. Data Reduction and chunking C. Event logs domain knowledge
9	Software engineers	A. Domain knowledge sharing B. Technical brainstorming
10	Functional engineer	A. Domain knowledge

Stakeholder analysis is a crucial step in designing any tool or system. In the case of a tool to automate event log file analysis, it is important to consider the perspectives and concerns of various stakeholders. One of the stakeholder concerns is false positive results, which may lead to the tool missing important anomalies or root causes. To address this concern, the tool could incorporate additional methods for identifying and flagging potential false negatives, such as manual verification by human experts or machine learning algorithms.

Another concern is keeping the tool up-to-date with changing environments and sequences. This requires ongoing maintenance and updates to ensure the tool remains relevant and effective. Stakeholders may also express concern about false outliers or the interpretation of results, which can be addressed through clear and transparent documentation and explanation of the tool's processes and outputs.

Verification is another challenge stakeholders may raise, as they will want to ensure the tool is accurate and reliable. This can be achieved through rigorous testing and validation processes, including both automated and manual verification. Window size can also be a problem, as it can affect the accuracy and usefulness of the tool's results. To address this, the tool could incorporate adjustable window sizes or adaptive algorithms to account for changing data patterns.

Interconnected data and large amounts of data can also pose challenges, as stakeholders may be overwhelmed by the sheer volume of information. To address this, the tool could incorporate data visualization techniques to make it easier to identify trends and anomalies. Stakeholders may also express a need for quick and accurate pointers to the root cause, which can be achieved through advanced algorithms or machine learning techniques.

Overall, stakeholder analysis plays a critical role in designing a tool to automate event log file analysis. By considering the concerns and perspectives of various stakeholders, the tool can be designed to meet the needs of all parties involved and provide valuable insights into the root causes of issues in complex systems.

8.5 Eindhoven University of Technology

This section describes the main stakeholders of the Eindhoven University of Technology. They are mainly responsible for ensuring that the quality of the project deliverables meets the EngD standards. The knowledge and experience of these stakeholders regarding the project business logic might be limited. However, they provide help and support from the academic point of view, if needed. Table 7 lists the main stakeholders and their concerns. Dr. Kruger was primarily involved in the project via PSG meetings. He got regularly updated either by emails or private meetings. Dr. Dajsuren occasionally participated in the PSG meetings and urgent situations meetings.

Table 9 TU/e stakeholders

Id	Role	Concerns
1	TU/e EngD ST Program Director	A. Ensuring that the quality of the project is following the EngD program standards B. Trainee's graduation C. Project Success
2	TU/e Supervisor	A. Monitoring the trainee’s progress B. Helping the trainee to overcome potential difficulties by giving directions in case of need C. Evaluating the project achievements
3	TU/e EngD Trainee	A. On-time graduation B. Project success C. Developing project management and soft skills D. Developing technical skills, including designing skills and computer vision E. Stakeholder expectation management

It is important to note that not all stakeholders have the same influence on and interest in the project. It is crucially important to figure out how each stakeholder could influence the project and its direction. During the project, the trainee tried to grab the attention of the stakeholders with less interest but high impact on the project by providing interesting results to increase their involvement to be able to request and get more specific information.

Appendix B. Project Management

Project management is one of the key factors in project success. It aims to define the project roadmap and strategies to tackle difficulties and risks within the project, facilitating the achievement of the project goal within the given constraints. In this project, project management was done using earned-value analysis. During the first weeks of the project, the main milestones were determined, and later on, during each iteration, more detailed tasks were planned. The following sections explain the way of working, project planning, and risk management in this procedure.

8.6 Project network diagram

A project network diagram is an important tool because it helps teams visualize the activities that need to be completed over the duration of a project. It also gives crucial contexts such as task duration, sequence, and dependency. As it is also shown in Figure 5, a project network is a graph that shows the activities, duration, and interdependencies of tasks within your project.

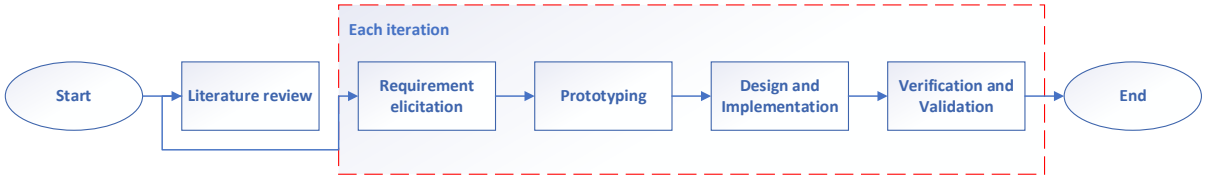

Figure 5 Project network diagram

About the Author

I am a skilled software engineer and machine learning enthusiast with a strong background in computer science. I have extensive experience developing software applications for a variety of industries, including supply chain management, health, and IT consulting and outsourcing. My passion for machine learning is demonstrated through my completion of advanced courses, participation in industry projects that involve data analysis and modeling, and the development of data-driven solutions to complex problems in diverse fields. Currently, I am pursuing an EngD program in the Netherlands to further expand my knowledge and skills in software development and machine learning. I am excited about exploring interesting opportunities to apply my expertise and further my career.