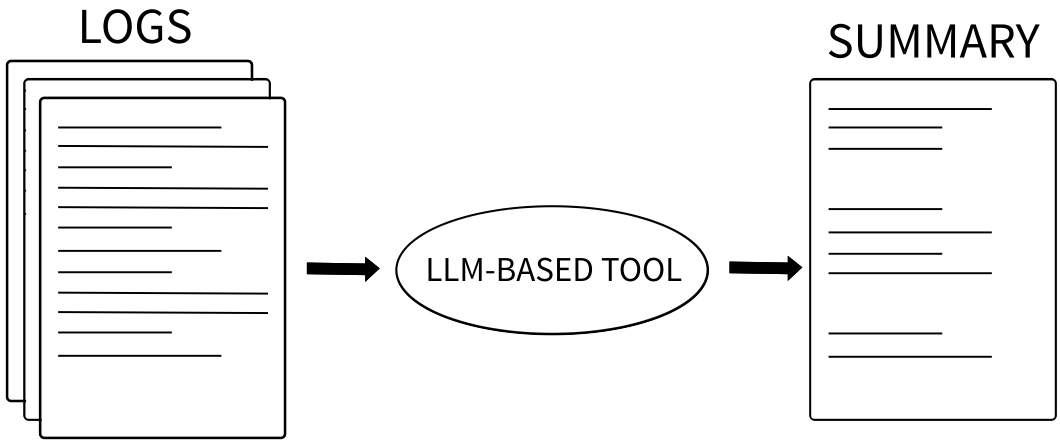
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



LLM-based Log Analysis for Fault Localization in the Automotive Industry

Master's Thesis in Computer science and engineering

Hampus Rhedin Stam
Anton Ekström

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's Thesis 2025

LLM-based Log Analysis for Fault Localization in the Automotive Industry

Hampus Rhedin Stam
Anton Ekström

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

LLM-based Log Analysis for Fault Localization in the Automotive Industry

Hampus Rhedin Stam, Anton Ekström

© Hampus Rhedin Stam, Anton Ekström, 2025.

Supervisor 1: Francisco Gomes de Oliveira Neto, Department of Computer Science and Engineering
Supervisor 2: Gregory Gay, Department of Computer Science and Engineering
Industrial Supervisor: Sabina Edenlund, Volvo Cars
Examiner: Philipp Leitner, Department of Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

LLM-based Log Analysis for Fault Localization in the Automotive Industry
Hampus Rhedin Stam, Anton Ekström
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This thesis investigates the application of large language models (LLMs) to aid practitioners of log analysis for fault localization in the automotive industry. An existing LLM-based log summarization tool is extended and evaluated, focusing on the cognitive load of practitioners and how satisfied they are with the tool. The effect of LLM-based log summarization on productivity of practitioners in the automotive industry is investigated through a case study at a company within the automotive industry. Think-aloud sessions and semi-structured interviews are carried out to asses the impact of the tool on the fault localization flow of study participants. Results suggest that LLM-generated log summaries can aid practitioners by giving them a first glance of the issue, thereby potentially reducing manual effort and improving productivity. However, the results also suggest that the context of the issue, domain knowledge, and interactivity of the tool plays a major role for success. A lack of context and means for the practitioner to guide the tool could result in a less effective workflow with higher cognitive load. The thesis provides insights on the integration of LLM-based log analysis tools within fault localization workflows in the industry, highlighting both the benefits and challenges of deploying LLMs in real-world fault analysis scenarios.

Keywords: Artificial Intelligence, Large Language Models, Software Log Analysis, Log Summarisation, Fault Localisation, Debugging, Automation, Automotive

Acknowledgements

First and foremost, we would like to express our deepest gratitude to our academic supervisors, **Francisco Gomes de Oliveira Neto** and **Gregory Gay**, for their guidance and invaluable feedback throughout the development of this thesis. Their expertise and support has been instrumental in shaping this work.

We extend our heartfelt thanks to our industrial advisor, **Sabina Edenlund** from Volvo Cars, for her support and the insightful guidance that she has provided. Her practical insights and encouragement greatly enriched this research.

We are also immensely grateful to our **friends and family** for their continuous love and support. Your encouragement has been a source of strength and motivation.

A special thanks goes to **the creator of Wiener Melange**, whose delightful creation has provided us with the energy and motivation needed to complete this thesis.

Thank you all for your contributions and support <3

Hampus Rhedin Stam & Anton Ekström, Gothenburg, June 2025

Contents

List of Figures

List of Tables

Glossary

AI-generated Refers to content, data, or analysis produced by artificial intelligence systems, often using machine learning algorithms to simulate human-like creation and reasoning.

fault localisation The process of identifying the specific location or cause of a fault within a system, often using diagnostic tools and techniques to facilitate repairs and improve system reliability.

LLM-based Refers to technologies or applications that utilize Large Language Models, which are advanced machine learning models trained to understand, generate, and manipulate human language.

LLM-based log analysis tool The tool used in this study that leverages Large Language Models to analyse and summarise logs, helping practitioners of troubleshooting to identify faults.

log analysis Pertains to the activity of reading logs from software systems or test executions for the purpose of troubleshooting.

the partner company The automotive company that collaborated in this research and where the study was done at.

Acronyms

AI Artificial Intelligence.

CI Continuous Integration.

DLT Diagnostic Log and Trace.

LLM Large Language Model.

NASA-TLX NASA Task Load Index.

OSB One Software Download Body.

SE Software Engineering.

1

Introduction

The process of fault localisation is often one of the first steps when debugging faulty software [45]. Before the developer can begin to fix the root cause of the issue they need to identify the location of the fault. Therefore, fault localisation plays an important role in the maintenance and development of software systems and is a critical step in ensuring the quality of the system. However, fault localisation can be time-consuming and labour-intensive [45, 18], especially with extensive logs and a large program environment. A systematic approach to fault localisation can lead to significant productivity improvements of the quality assurance process, which makes research contributions in the subject area valuable.

In modern software development practices, it is common to monitor the status and health of a system by logging messages containing relevant information about its state [26]. There exist many sophisticated solutions for producing readable and informative log messages, such as log4j[1] or spdlog[2]. In its simplest form, logging is performed by inserting logging statements into the source code of the system. The logs may be sent to the standard output of the program so that the information can be captured and saved in a log file. The log files can later be analysed to gain further insights in case it is discovered that the system encountered an error or failure during its execution [22]. Log files are produced whenever the system is executed: in a production runtime or during the execution of test cases. Log files provide a comprehensive record of system activities and events, which can serve as a valuable indicator for pinpointing the source of failures. By systematically examining the log messages, one can trace the sequence of events leading up to an issue, thereby facilitating fault localisation and root cause analysis.

Log analysis is a common activity during fault localisation of complex software systems [22]. It can be especially useful to use log files for long test executions spanning multiple test cases, since these tend to be even more difficult to debug effectively. For example, there may be complex interdependencies between test cases that cause the order of test case execution to affect the results of the tests. To understand such an issue comprehensively, we are required to view the test execution as a whole, for which log analysis is a suitable technique to begin the process with.

[1] `https://logging.apache.org/log4j/2.x/`
[2] `https://github.com/gabime/spdlog/`

As software systems become increasingly complex , the logs produced by the systems also grow in size and complexity [26]. When a failure occurs in a CI pipeline, an extensive workflow is required to identify the nature and location of the fault in order to write a fault report for the appropriate team of developers, which will then address the issue. The test engineer must first examine which test cases have failed or encountered errors. As a consequence of increased system size, the task of analysing the logs becomes more challenging and laborious.

For the largest and most complex software systems, the log analysis process can quickly become overwhelming, such can be the case for the automotive industry. For example, it is time-consuming to analyse the logs from a complete vehicle assembly. Such a system may produce huge quantities of log messages and log files for a single test suite execution, which is primarily the reason as to why the task is so time-consuming. If the fault is severe enough, or if there are too many faults existing at once, the complexity of fault localisation may cause delays in releasing new versions of the software.

1.1 Problem Statement and Proposed Solution

Pre-trained large language models (LLMs) are being applied to numerous problems in the area of Software Engineering (SE) research. As such, LLMs have recently emerged as a potential alternative to many of the traditional techniques within fault localisation [43, 20], software testing [41], and SE research in general. Fault localisation has witnessed LLMs being effectively implemented to perform tasks such as test case generation [34], code completion [19], fault mitigation [46], and generating incident reports [1], just to name a few. However, in comparison to areas pertaining to code generation, there are relatively few studies on fault diagnosis and code repair [41].

Log analysis entails reading a large volume of log messages and processing this information to draw meaningful conclusions. This task is similar to the capabilities of large language models (LLMs) [49], which are well suited for understanding and generating natural language text. Recent advances in large-language models (LLMs) have prompted researchers in many areas of SE to investigate whether those models may be used to replace or enhance previous manual methodologies [41].

Since software systems are growing in size and complexity, there is a growing need for tools that can aid in automating certain steps of the fault localisation process [45]. Software testing as a whole represents a substantial portion of the total cost of software development [2], indicating that large costs can be saved by improving productivity [7] in certain areas of software testing, such as fault localisation and log analysis.

We hypothesize that LLM-based tooling for fault localization and log analysis could potentially lead to developers feeling more satisfied by helping to reduce the time they spend on repetitive or boring tasks. Additionally, reduction of friction in the

fault localisation workflow may potentially help practitioners enter a state of flow [10]. Lastly, cognitive load [17] is likely to stay relevant as developers are having to maintain increasingly complex systems [45], and automation of mentally taxing activities in fault localisation workflows such as log analysis could potentially reduce the cognitive load of practitioners. Therefore, LLM-based log analysis could increase productivity [40, 7], because the factors cognitive load, user satisfaction, and flow are dimensions of productivity [16, 25].

Notoriously, log files exist in many different formats, and proprietary formats are commonly used in the industry [22, 26]. Due to the lack of standardization in log formats, it has been difficult to create a solution that works generally for any log type. Previously, log analysis has required us to employ different solutions depending on the structure and format of the logs, or develop complex techniques and domain-specific languages to describe the format of logs [5]. This problem unnecessarily causes work to be duplicated for many of the proprietary or niche log formats that exist in the industry, likely contributing to higher costs for software testing. We believe that the general-purpose nature of LLMs could accelerate the implementation of supportive tooling for log analysis. Widely used log formats may already have a substantial presence in the training data of LLMs, while examples of lesser used and proprietary formats could be used for in-context learning [13, 24].

Additionally, the study focuses on logs that are produced during execution of test cases. Other logs, such as those produced for the purpose of providing analytics or anomaly detection on systems deployed in production environments are out of scope for this study.

Our study is mainly focused on log summarization, and is not aimed at automating the entire fault localisation process. Other summarization tasks such as fault reports or source code summaries is not the focus of this study. The LLM-based approach in this study is supposed to aid the developer in their fault localisation process, by providing generated summaries of software system logs.

1.2 Research Objective

In our thesis, we conducted a case study at a company within the automotive industry to explore the usage of LLMs for log analysis in an industrial setting. The study was conducted through a partnership between the company, Chalmers University of Technology[3], and Software Center[4], with the common goal of sharing insights in industrial digitalization and software engineering [36]. We were able to extend and evaluate a prototype LLM-based log analysis tool, created by a small team of developers at the partner company, that automatically generates summaries of log files. Prior to our study, the tool has not been used in large scale and only a few test engineers have used the tool in practice.

[3] `https://www.chalmers.se/`

[4] `https://www.software-center.se/`

The study aims to examine the impact of automatically generated log summaries on the fault localisation workflow of practitioners, focusing on log analysis. We devised three research questions, outlined in Section 4.1 with the purpose of exploring whether LLM-based techniques are able to: effectively aid practitioners reduce cognitive load, improve satisfaction, and facilitate better flow in their fault localisation workflow. We conducted think-aloud sessions with 10 participants from the partner company, each session followed directly by an assessment of cognitive load and satisfaction level with a semi-structured interview. We gathered insights based on qualitative data using thematic analysis of the interviews, complemented by quantitative measures of cognitive load.

1.3 Contributions

The practical implications of this thesis bring concrete findings on how the LLM-based log analysis tool affects the workflow and productivity of practitioners, to support and inform initiatives on improving LLM-based tooling within the partner company, the automotive industry, and the software development industry as a whole. For instance, practitioners using the summaries requested more interactivity with the tool, and to be able to work iteratively throughout the fault localisation process. The usefulness of the summaries were also considered to be affected significantly by their ability to influence to output of the tool, such as generating new summaries with added context about the issue. Furthermore, the recommendations we present can be of use to companies and studies seeking to advance LLM-based tooling, by providing directions on aspects of such tools to focus on improving.

As scientific contributions of this thesis, we provide empirical evidence to two areas of LLM research, for which there exist relatively few studies: LLM-based log analysis and human-AI interaction. Due to the field of LLM research being rather new, we focus on gathering explorative findings to support future studies. Our exploration of how the LLM-based log analysis tool may affect the fault localisation workflow of practitioners brings attention to human-AI interaction — an important aspect of LLM-based systems — that can often be overlooked in the search of optimizations and performance improvements. Our contributions can support future studies in further investigating the subject, focusing, e.g., on how AI can aid humans in their work.

2

Background

This chapter describes concepts that are relevant to the study. It begins with describing software testing and fault localisation. Then it continues with explaining productivity in software development and presenting frameworks which can be used to define developer productivity. The frameworks are used to connect the study to developer productivity, and helps motivating how the LLM-based log analysis tool may affect the productiveness of practitioners. Finally, a description of the large language models and the LLM-based log analysis tool itself is presented.

2.1 Software Testing

Software testing is the process of validating whether a system meets its specification through executing the program and inspecting whether the output of the program or parts of the program matches the expected behaviour [42, 32]. Tests are usually divided into test cases, and organized into logical groups of test suites. A test case usually contain an input and an expected output. Software tests are divided into several levels of granularity that usually follow the V-Model [6], but can vary between system domains. In the V-Model, system tests progress from low-level tests (e.g.. Unit tests) to more complex high-level tests. Unit tests tend to be seen as the smallest type of individual test cases, in which the unit test is only concerned with a single module [32]. A module is a self-contained unit that is responsible for a single function. It is common for low-level unit tests to make up the majority of the tests for a system [28]. Integration tests are higher level than unit tests, and involves testing how multiple units interact, focusing on the communication between modules [42]. System tests are the highest level, in which the complete system is tested.

2.1.1 Software Testing in the Automotive Industry

Similarly to other automotive companies [6, 37], and as suggested by safety regulations in the automotive industry, the partner company follows a modified V-Model, defining four levels of testing granularity: unit, component, domain, and complete.

Figure 2.1 shows a simplified flow that describes how failing tests are handled at the partner company. Tests on the unit-level are usually small enough to be mitigated

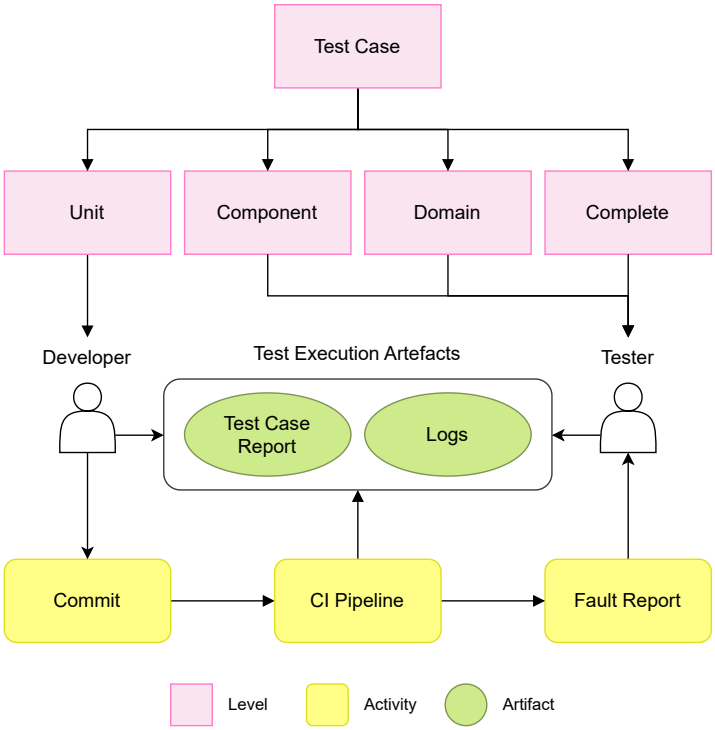


Figure 2.1: The tester or developer receives an issue based on the granularity of the test. The tester creates a fault report for the developer, while the developer themselves creates a fix for the issue.

by the developer themselves, in the team that is responsible for the module. For higher level of abstraction it can be difficult to pinpoint where the issue is located, and a specialized tester whom has a greater understanding of the system as a whole is then responsible for locating the fault and assigning a fault report to the appropriate team. The logs from the test case is used by both developers and testers. However, testers are the focus of this study since they tend to be the most involved with logs.

2.2 Fault Localization

Fault localisation aims to identify the location of suspicious entities (modules, functions, statements, etc.) that may cause the program to fail or to exhibit unexpected behaviour [45, 3]. It is often relevant to inspect data related to the issue, for example: quality gates, specifications of the test case, issue reports, source code, or logs.

The simplest methods of fault localisation involve the developer manually adding logging statements or assertions to the source code [45], or intuitively troubleshooting with the help of debugging techniques such as adding breakpoints or using profiling tools [45]. As other studies have noted, these methods of fault localization are both time-consuming and tend to heavily rely on the developer's experience of troubleshooting within the domain of the issue [24].

The process of fault localization may involve other activities such as anomaly detection, failure diagnosis, and root cause analysis. Sometimes a preceding step to fault localization, anomaly detection involves identifying potential faults, or otherwise unexpected behaviour in the system [18, 14]. While fault localization aims to find the location of a fault, the purpose of failure diagnosis and root cause analysis is to perform a deeper investigation of the events leading up to the issue and identifying the potential root cause [18].

2.3 Log Analysis

The analysis of logs is an activity often carried out as a step in locating, detecting, and investigating faults in software systems [18, 22]. Before applying automated log mining techniques, it is often necessary to convert semi-structured logs into a structured format — which is known as log parsing [18]. In conjunction with log parsing, pre-processing is optionally performed to omit unnecessary information, replace variables, or otherwise modify the contents of the log. There are numerous methods of mining the structured log data for insightful information, such as statistical methods [18], clustering methods [9], or machine learning methods [14].

2.3.1 Log Summarization

Log summarization is the process of condensing log data to highlight the most crucial information, making it easier for engineers and analysts to understand and act upon

the system's behaviour. Logs can be extensive and cognitively demanding to go through, summarization helps in extracting meaningful insights without needing to parse through every single log message manually [15]. This can be very helpful in large systems where logs can be voluminous and complex, and can potentially accelerate the fault localisation process.

2.3.2 Log Formats

Log files capture messages that are produced during the runtime of a software system [5]. Log files may come from a live production environment, but may also be created after executing a test suite consisting of one or more tests. There is a lack of standardization of log formats [22], and proprietary log formats are common in the industry. Even though many companies are using proprietary formats, logs usually follow a similar structure. Messages are ordered in chronological order, sorted after timestamps, and delimited by line breaks. A message may contain certain metadata such as timestamp, log level, or a location in the source code.

2.4 Productivity in Software Development

Productivity in software development can be defined and interpreted in multiple ways. The DevEx [25] and SPACE [16] frameworks are used in the study to describe and motivate how cognitive load and satisfaction level may contribute to the productivity of practitioners. For example, both frameworks state that 'flow' is a factor that affects productivity. It is also apparent from studies that productivity is not easily defined and there is no single quantitative metric that captures the concept as a whole. We attempt to make connections to productivity using these frameworks, but our research questions do not directly cover the topic, since we believe this is better suited for a future study.

2.4.1 SPACE

The SPACE framework describes five dimensions that may affect productivity [16]. The five dimensions are:

- **S**atisfaction and well-being
- **P**erformance
- **A**ctivity
- **C**ommunication and collaboration
- **E**fficiency and flow

We are focusing on two dimensions, as we deemed these to be most feasible for us to study: satisfaction and well-being, efficiency and flow. Our study addresses satisfaction level directly as one of the research questions. Satisfaction is correlated

with productivity [16], indicating that if practitioners are overall satisfied with the tool it is more likely to impact their productivity in a positive manner. Efficiency and flow is covered by the last research question which investigates the impact of the tool on fault localisation workflows in the industry. Reduction of repetitive or time-consuming activities caused by the tool may help practitioners achieve flow state.

2.4.2 DevEx Framework

The DevEx framework is a developer-centric approach to measuring productivity [25]. Unlike traditional productivity measures, which often emphasize output or time-to-completion, the DevEx framework considers the qualitative aspects of a developer's work environment and workflow. In order to measure productivity, the framework considers three dimensions — feedback loops, cognitive load, and flow state.

Feedback Loops: Focuses on speed and quality of responses for tasks that relies on feedback. Slow feedback loops can result in frustration or delays. Faster feedback loops minimises the friction for developers to complete their tasks. Feedback loops can measured by, e.g., satisfaction with time it takes to validate a local change or time it takes to for CI jobs to finish.

Cognitive Load: Encompasses the mental effort required for developers to perform a task. A high cognitive load can be an indicator of complex codebases, poorly documented systems or lack of consistency. Lower cognitive load allows developers to focus more on problem-solving and less on managing complexity.

Flow State: Refers to how enabled the developers are to enter a mental state to be fully engaged in their work, often described as being "in the zone". A low flow state means that developers may have a hard time to focus and can not fully emerge in their work. To improve the flow state, interruptions and delays should be minimised. Flow state can be measured by perceived ability to focus and avoid interruptions or frequency of unplanned tasks or requests.

2.4.3 NASA Task Load Index

NASA Task Load Index (NASA-TLX) is a multi-dimensional assessment method designed to collect a subjective workload measurement for performing a specific task [17]. The framework divides productivity into six different dimensions, described in Table 2.1. Our study frames cognitive load in the context of productivity, motivated by the DevEx and SPACE frameworks, which is further measured using measured using NASA-TLX.

The framework helped us to come up with metrics for the study to compute measures for productivity. Regarding cognitive load, we used the NASA Task Load Index framework to assess a numeric value that could be used to compare different data points to each other.

Table 2.1: Each dimension that is assessed for NASA-TLX with the endpoints of the range as well as a description of what the rating scale entails [17]. Adapted with permission.

RATING SCALE DEFINITIONS		
Title	**Scale**	**Descriptions**
Mental demand	Low/High	How much mental and perceptual activity was required (e.g., thinking, deciding, calculating, remembering, looking, searching, etc)? Was the task easy or demanding, simple or complex, exacting or forgiving?
Physical demand	Low/High	How much physical activity was required (e.g., pushing, pulling, turning, controlling, activating, etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?
Temporal demand	Low/High	How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?
Own performance	Good/Poor	How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?
Effort	Low/High	How hard did you have to work (mentally and physically) to accomplish your level of performance?
Frustration	Low/High	How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task?

2.5 Large Language Models

Large language models (LLMs) are generative Artificial Intelligence (AI) models that produce text in natural language by predicting the probability of the next token in a sequence of words. A token is a basic unit of text, which can be as small as a single character or as large as a word or subword, depending on the tokenization method used. Essentially, tokens are the building blocks that the model processes to understand and generate text.

LLMs are trained on large datasets consisting of books, news articles, forums, blogs, and other forms of textual content often retrieved from the internet. A user can ask and converse with an LLM through different prompts. Due to the large amount of source code available from websites such as GitHub[1], LLMs are also capable of helping with tasks related to software engineering tasks, such as program repair [46] or fault localisation [20].

2.5.1 Token Limit

The context window of an LLM is the number of tokens that the LLM is able to hold in memory at any given moment [15]. For a conversational model, the entire conversation of messages back-and-forth needs to fit within the context window for the LLM agent to be able to remember the conversation. The same principle applies to log summarization, the entire log file needs to fit within the token limit of the LLM. Otherwise, the model may miss important information in the file. Additionally, the ability of the LLM to retain information tends to suffer when nearing the token limit, leading to worse performance and more hallucinations [29].

Since the emergence of transformer-based models [39], LLMs have gotten increasingly capable of handling large volumes of data. Models such as GPT4o[2] have a context window of 128 000 tokens and are capable of generating solutions to software development problems with the ability to reason about complex topics [27].

2.5.2 Prompt Design

Prompts are often classified as either zero-shot or few-shot prompts [21, 24]. Prompts that are classified as zero-shot only contains instructions on how to produce the expected output, but does not provide any examples. Few-shot prompts however, include examples of the input data and what the appropriate output is.

Chain-of-thought (CoT) refers to prompting the model with intermediate reasoning steps. Few-shot CoT involves providing the model with examples of step-by-step reasoning. Zero-shot CoT may simply be adding "Let's think step by step" to the prompt, which has shown to improve the results on several benchmarks [21].

In-context learning refers to providing the LLM with information in the prompt that

[1] `https://github.com`
[2] `https://openai.com/index/hello-gpt-4o/`

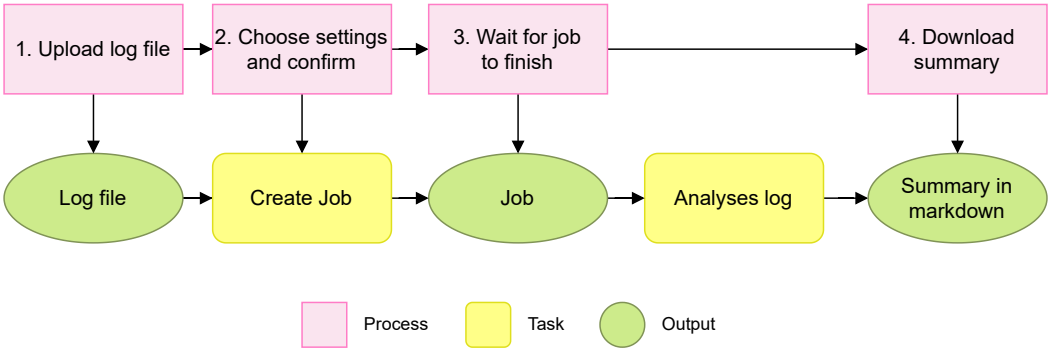


Figure 2.2: Workflow of a user that uses the LLM-based log analysis tool. The user uploads a file, waits for it to finish, and then downloads the resulting summary. The upper row represents the steps that the user is seeing.

helps the model produce a better output.

2.6 LLM-Based Log Analysis at the partner company

The partner company recently implemented an LLM-based log analysis tool that initially supported only the log format Diagnostic Log and Trace (DLT), which contains messages about the status of the system during execution. The tool uses GPT-4o[3] (version 2024-08-01-preview) by OpenAI to summarize data. As illustrated in Figure 2.2, the workflow starts by uploading a log file to be summarised through the web application. The user can optionally apply filters and customize the instructions for the LLM, such as a simple description of the error, or keywords to filter messages based on. The user then waits for the tool to finish processing the log file and downloads the Markdown[4] summary. An example of the summary can be seen in Figure 2.3. The LLM is been instructed to highlight the most relevant messages in the log file, and to provide suggestions on how to further investigate the issue.

2.6.1 MapReduce

The tool is based on the MapReduce framework, as it is an efficient method for processing large datasets in parallel, which the tool benefits from by summarizing several chunks of the log simultaneously.

MapReduce is a programming framework that utilises parallel processing to speed up the processing of large datasets [11]. The model encompasses two functions called "Map" and "Reduce". The Map function chunks the data into smaller parts,

[3] https://learn.microsoft.com/en-gb/azure/ai-services/openai/concepts/models?tabs=global-standard%2Cstandard-chat-completions#gpt-4

[4] https://commonmark.org/

```
## Summary
- The log contains entries from two distinct sessions:
    - **Session: ECU1**
        - Error indexed at [62]
    - **Session: ECU2**
        - Error indexed at [64]
- Both sessions reported the same error 32 times, classified as 'error'
type.
- The error description in both sessions indicates a failure in the
`ClientImpl:ConnectAgent` process.
- The specific cause of the error is "connect failed: No route to host,
endpoint: MASK.1.2.3:MASK", suggesting a network connectivity issue.

## Suggestions
- Investigate the network settings and route configurations for both
sessions to identify and correct any incorrect routing or
configurations.
- Ensure that the host's network is properly set up to allow connections
to the endpoint specified (MASK.1.2.3:MASK).
```

Figure 2.3: Example of a summary that has been generated by the LLM-based log analysis tool. Instructions to divide the response into a summary and suggestion section are given to the LLM in the prompt.

converting them to key/value pairs. Chunks are mapped in parallel to speed up the process. The Reduce function then processes the results from the Map function. The Reduce function could merge the data from the Map function or perform further operations on it. The Reducer returns the output of the model. Figure 2.4 shows an example of how a book could be summarized using MapReduce. In the tool, the LLM is used in both the map and reduce stages, first to summarize each chunk and then to combine all chunks into one summary.

2.6.2 Internal Workflow

To begin with, the file is sent through a pre-processing stage. Afterwards, the log file is split into chunks that are processed in parallel. The last step is to reduce the chunks into a final summarization that is presented to the user. The dataflow of the tool is illustrated in Figure 2.5, which shows the sequence of steps that are described in the rest of this section.

Pre-processing: Log files are pre-processed to remove unnecessary information. The log is formatted to remove certain characters in order to reduce size and streamline the information. The log is also divided into segments, using the separator tokens START and END. One segment consists of messages that corresponds to the same session, an example can be found in Figure 2.6.

Chunking data: The logs are split into chunks. The chunking is determined by the available context window and tries to fit as many segments as possible.

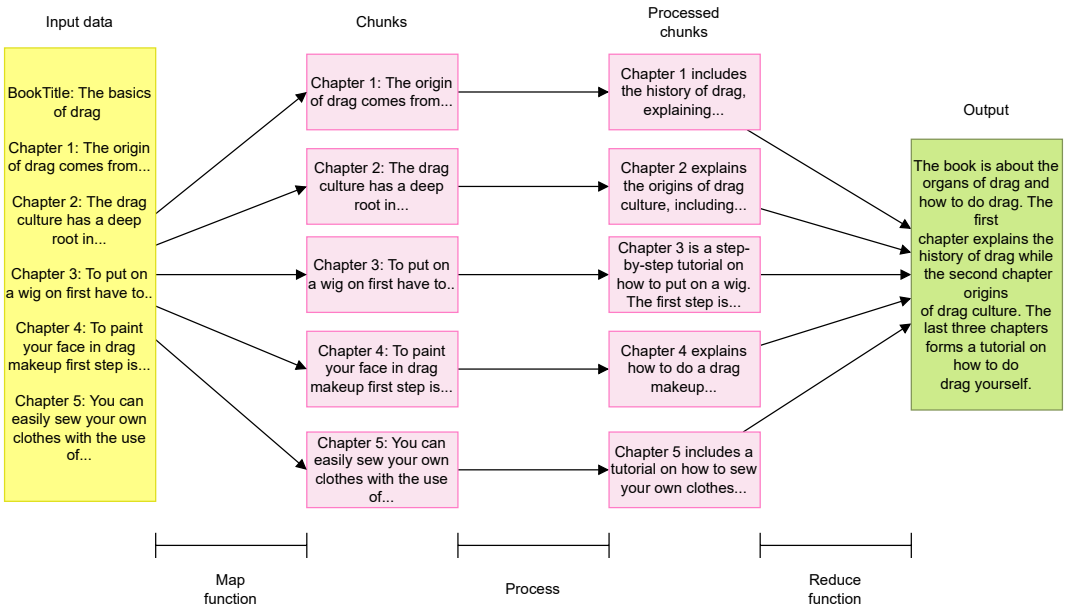


Figure 2.4: A visualisation of the MapReduce programming model. The Reduce function chunks the book into chapter and summarises each chapter in parallel. The Reduce function then uses the summaries of each chapter to create a summary of the whole book.

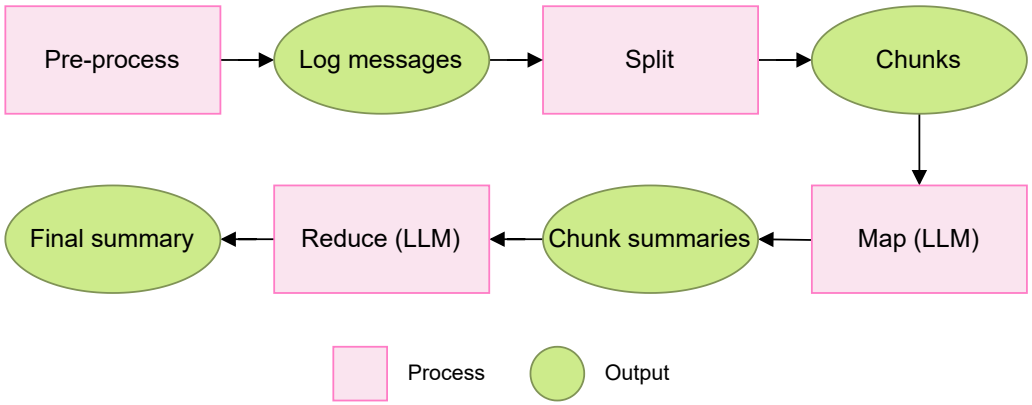


Figure 2.5: An overview of the steps involved in summarizing a log file. Green boxes show the steps in processing the data, represented by yellow ellipses.

```
START ABC-NNNN-NNNN-12345
[790] 2 error: set resets from XX failed for:
[791] 1 error: NN-NNN
[793] 1 error: NN-NNN
[44794] 352 error: Lost receive id: MASK
END
START ABC-NNNN-NNNN-12345
[794] 1 error: NNNNN are disabled for this platform
[795] 1 error: No NNN NNNNN enabled in platform
END
START ABC-NNNN-NNNN-123456
[812] 1 error: Can't parse NNNNN,impenable
[813] 1 error: open: imp parse failed, skipping imp
END
```

Figure 2.6: Example of how a log file looks after formatting. Blocks are delimited by lines with 'START' and 'END'.

Summarizing chunks (Map): This is the first stage in MapReduce. Each chunk is embedded in a prompt and sent off to a new LLM instance with a fresh context. Chunks are processed in parallel and isolation; there is no information shared between chunks while summarizing them. The tool uses a prompt template that instructs the LLM to summarize the most important information in a chunk.

Combining chunks (Reduce): When all chunks have been summarized independently, the reduction step combines all of them into a final summary. All the summarized chunks are embedded in a prompt, and the LLM is instructed to summarize the information it is given.

2.6.3 Prompts

The prompts are designed with zero-shot prompting, since the prompt is giving an example of the formatting and some instructions, but does not give an example of appropriate responses to example inputs. There are two prompt templates that vary slightly based on the log type. The first prompt is used in the mapping stage for summarizing the individual chunk (Figure 2.7), and the other prompt is used for reducing the chunks into one summary (Figure 2.8). The prompt template begins with general instructions and context, which is followed by an example of the log format. Finally, there is a list of precise instructions, before giving the input data at the end.

```
You are a senior software engineer. Your task is to summarize logs. The
logs given to you are divided into sessions following this format:

```
START <session title>
[<index>] <occurrence count> <log type>: <payload>
END
```

Log to be summarized: <LOG CHUNK>

Use the session title when pointing to a specific session. Mention the
index when pointing to a specific message.

Draft the summary following these steps:
1. Read and understand the relation among the occurrence count, log type
and the payload.
2. Point out the important errors in the log and identify potential root
causes of the errors.
3. Organize information and write a concise summary. Avoid using vague
or overly general words.
4. Revise to remove any redundant information.
Summary:
```

Figure 2.7: Example of the log summarization prompt template for the map stage.

```
You are a senior software engineer. Your task is to summarize logs.
Given the following summaries of different parts of the same log,
combine them into a concise summary.

Provided summaries: <CHUNK SUMMARIES>

Draft the summary following these steps:
1. Identify common points among the summaries. These will be the
backbone of the final summary.
2. Point out unique information in each summary. These could be
important details.
3. Organize information and write a concise summary. Avoid using vague
or overly general words.
4. Provide suggestions to resolve the errors. Be specific and
actionable.
5. Write the final summary in formatted Markdown.
Example:
{{
\#\# Summary
- Problem 1
    - Session 1 (index 123)
    - Session 2 (index 456)
- Problem 2
\#\# Suggestions
- Suggestion 1
- Suggestion 2
}}

Final Summary:
```

Figure 2.8: Example of the log summarization prompt template for the reduce stage.

3

Related Work

This chapter presents an overview of related studies that are vital to understanding the wider context of the study. Traditional methods such as statistically-based and machine learning-based are introduced first, followed by LLM-based methods. The chapter ends with discussing how our thesis builds upon, and differentiates itself from the other studies.

3.1 Traditional Approaches to Fault Localization

Advanced traditional approaches to fault localisation are based on complex algorithms to provide the user with a list of potentially faulty components ranked in order of suspiciousness, an example is the statistically-based DStar method which is based on binary similarity coefficient analysis [44]. Other commonly used approaches are slice-based, model-based, spectrum-based, and machine learning-based [45]. There are several types of software faults that may be targeted, for example, some techniques are for locating functional faults while other techniques diagnose performance and security vulnerabilities [45]. Fault localisation techniques may use a wide range of input data, such as results from test cases [45, 44], source code, fault reports, or logs [45, 4].

3.1.1 Traditional Approaches to Log Analysis

Log analysis encompasses several techniques for locating faults, analysing the root cause of failures, or detecting faulty system behaviour [18, 22]. Traditional methods for anomaly detection and fault localisation using log analysis include: dimensionality reduction, clustering, classification, frequent pattern mining, graph mining, and machine learning [18, 4].

LogFaultFlagger is a machine learning-based log analysis method for fault prediction and localization [4], it employs a step in which the log messages in failing logs are subtracted from those of passing logs to select messages that are more likely to be related to the issue. LogLens [12] and DeepLog [14] are also machine learning-based methods for anomaly detection, both claiming to outperform older methods based on traditional log-mining techniques. FDiag is a statistically-based log analysis method for diagnosing the root cause of failures using logs [9]. LogCluster is a

clustering-based log analysis method for fault localisation [23].

3.2 Automating Fault Localization using LLMs

The recent surge in popularity of LLMs has inspired studies to apply LLM-based methods for fault localization and anomaly detection [24, 15, 20, 47]. A notable advantage of LLM-based methods is the opportunity to provide interpretable results in the form of natural language explanations and motivations, as opposed traditional methods usually only providing a simple score representing the suspiciousness of a certain piece of code [24, 20].

3.2.1 LLM-based Fault Localization

The LLM-based fault localisation technique AutoFL is able to autonomously retrieve relevant code segments and generate explanations for faults [20], and in some cases performed better than traditional state-of-art techniques like DStar [44].

BugBlitz-AI is an LLM agent that automates result analysis and bug reporting by utilizing several modules specialized in fault localisation, fault summarization and duplicate error detection [47]. BugBlitz-AI employs multiple LLM agents that specialize in different tasks and communicate with one another.

3.2.2 LLM-based Log Analysis

LogPrompt is an LLM-based log analysis method using zero-shot prompting which has shown promising results of outperforming previous approaches by wide margins, without requiring any in-domain training [24]. LogPrompt performs anomaly detection, log parsing, and log interpretation. The study also showed that certain prompt strategies enhanced the performance of the LLM significantly, compared to simpler prompts. The authors of LogPrompt suggests potential use for industrial applications.

3.2.3 LLM-based Summarization

Summaries of news articles generated using LLMs have found to be on par with human-written summaries [48]. The study notes that instruction tuning (prompting) had a tangible impact on the quality of the zero-shot summarizations.

Bug report summarization has been investigated by several studies [30, 47] to help with duplicate report detection for bug reports. The generated report summaries were found to be helpful for developers that were performing duplicate report detection, and most developers preferred working with the summaries as opposed to entire bug reports [30]. LLM-based summarization of source code to describe the functionality of software systems has been shown to surpass state-of-the-art summarization methods, while not quite reaching the level of human-written summaries [30]. However, studies have also expressed concern over the factual correctness of

the generated text [38], finding that automatic metrics may not correlate with the actual quality of the summary.

3.2.4 LLM-based Summarization of Logs

To the best of our knowledge, only few studies on LLM-based log summarization, however, an early exploratory study shows there is potential for LLM-based log summarization to surpass previous state-of-the-art techniques [15]. The method proposed in the study uses a sliding window to sequentially process chunks of log messages, effectively circumventing the context window of LLMs by keeping an in-context memory. This method allows for processing much larger log files.

3.3 Research Gap

Generative tasks using LLMs such as test case generation, test input generation, and code repair has gathered significant traction in SE research [41]. Meanwhile, the application of LLMs for troubleshooting activities such as root cause analysis, fault localization, and log analysis remains less explored. Recent studies have shown that LLM-based methods tend to suffer from hallucination and generating low quality output [35], however, these methods still compare well to traditional techniques. Since there is much left to be desired in terms of accuracy and consistency, we hypothesize LLMs may be better suited for methods involving humans in the feedback loop. Therefore, we believe LLMs can be used to greater effect by helping developers to interpret and analyse logs.

A previous study used a sliding window technique to keep memory of previous log messages in context [15]. Similarly, the tool we are studying uses the MapReduce framework to process chunks of log messages individually, and then reducing the summaries one by one into a final summary of the entire log. A potential benefit of the approach studied in our thesis is that several chunks can be processed by different instances of LLMs in parallel. Either way, it may be relevant to compare and contrast both approaches for deciding on an appropriate design for future implementations.

Our contributions are focused on qualitative insights from practitioners in the industry, as opposed to comparing the performance of the tool against other methods. Since our study is more human-centred and focuses on qualitative insights, we argue that our contributions provide an important perspective that similar studies on the subject may not explore as much. Hopefully, the insights from this study can complement other studies and lead to a more nuanced and varied body of knowledge for future studies to build upon.

It has been suggested that LLM-based methods for log analysis perform significantly better than traditional methods for unseen data and logs in new domains [24]. Since there also is a lack of standardization for log formats [22], we see great potential for LLM-based methods. We provide our findings in the context of a case study within the automotive industry, and focus on exploring how pre-trained LLMs are

able to contribute meaningfully for developers despite the lack of in-domain training on proprietary log formats and specialized problem solving activities.

4

Methods

This chapter describes the methods used to investigate the research questions. We begin by presenting the research questions themselves followed by further explanation of the context of the case study. Moreover, we explain the process of extending an LLM agent to summarise log files. Lastly, we describe the methods of sampling participants, data collection and analysis.

4.1 Research Questions and Overview

We explored the impact of LLM-based log analysis on the workflow of practitioners. Particularly, we aim to answer the following research questions:

> **RQ1:** How does LLM-based log analysis affect the cognitive load of practitioners?
>
> **RQ2:** How does LLM-based log analysis affect the satisfaction level of practitioners?
>
> **RQ3:** How does LLM-based log analysis impact existing fault localization workflows?

The first research question **RQ1** focuses on the extent to which log summarization may affect the cognitive load during log analysis, which can be associated with parsing, learning, and retention of information from the logs. Frameworks such as DevEx (Section 2.4.2) consider cognitive load to be an important factor to consider when assessing the productivity of developers. For this research question, the cognitive load of participants was assessed based on NASA-TLX, after they analysed logs with and without the help of the LLM-based log analysis tool.

The second research question **RQ2** assesses how the satisfaction level is affected of practitioners, from a manual workflow to workflow including the LLM-based log analysis tool. The perceived benefits by users play a significant role in the adoption of the tool. If there is no perceived value to be gained from using the tool, practitioners are less likely to bother using it. Practitioners that are satisfied with the tool may also be more prone to recommending the tool to their colleagues. The SPACE framework, described in Section 2.4.1, considers how fulfilled, happy, and healthy the practitioner of a task is a key dimension to fulfil productivity. A higher satisfaction

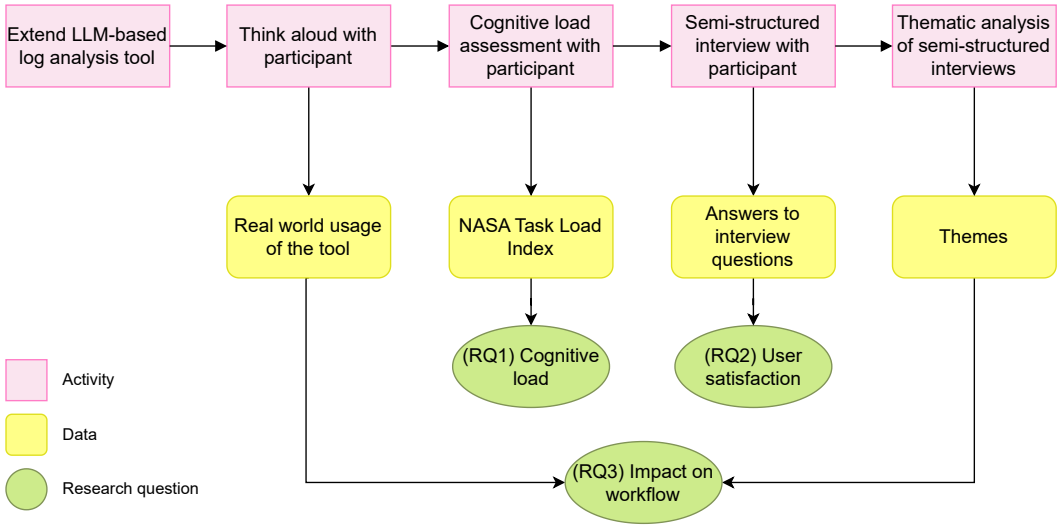


Figure 4.1: Overview of how the research questions relate to the main activities in the study.

of the worker means a higher productivity. For this research question, we gather qualitative information on satisfaction level through questions in semi-structured interviews with practitioners.

The third research question **RQ3** explores how the workflow of practitioners may be affected by introducing the LLM-based log analysis tool. We investigated how the tool might fit into their existing workflow, and if the workflow is altered in any significant way. We also investigated the benefits and drawbacks of the tool, considering if the tool comes with any unexpected changes to their workflow. For this research question, we perform thematic analysis to find patterns among the answers of study participants in the semi-structured interviews, and a think-aloud portion of the sessions provide further context to the themes.

Data was gathered through the use of think-aloud sessions and semi-structured interviews. The think-aloud session protocol can be found in Appendix A and the interview instrument can be found in Appendix B. The NASA-TLX measurement was collected in conjunction with the think-aloud sessions to assess cognitive load of participants during the session. We performed a thematic analysis for the semi-structured interviews to gather qualitative insights from the participants. The context of the case study is framed using terminology and guidelines by Runeson and Höst [33].

Figure 4.1 illustrates the main activities conducted in the study. The figure shows how the subsequent steps after the think-aloud activity are used to collect further insights, and then used to assess cognitive load, satisfaction level, and impact on the workflow, and thus answer the research questions.

Figure 4.2 illustrates how the LLM-based log analysis tool influences the data we

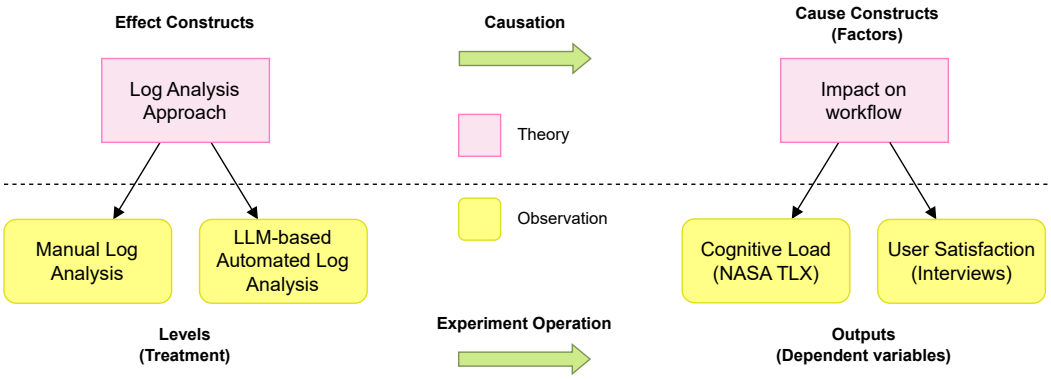


Figure 4.2: Illustrates the cause and effect relationship of the study constructs. The left side of the figure shows the two approaches, and the right side shows the effect those approaches may have.

Table 4.1: Plan for the case study, outlining the basic elements of the study [31]. We used the guidelines of Runeson and Höst as a guiding perspective for the study [33].

Objective:	Study the impact of LLM-based log analysis on fault localization in the automotive industry
The case:	Evaluating a LLM-based log analysis tool at the partner company
Theory:	Fault Localisation and Log Analysis
Research questions:	RQ1, RQ2, RQ3
Methods:	Think-aloud sessions, Semi-structured interviews
Selection strategy:	Convenience sampled practitioners at the partner company

collected for the research questions. The log analysis approach impacts the workflow and may affect cognitive load and satisfaction level. Some of the cognitive burden on the user could be alleviated if the generated summaries helped them localize the fault more effectively. The satisfaction level is related to the perceived value of the LLM-based log analysis tool by the user.

4.2 Context of the Case Study

The case study was conducted in partnership with an automotive company. Table 4.1 provides an overview of the case study, outlining how think-aloud and semi-structured interviews were used to evaluate the tool.

Automating the process of log analysis is relevant to the partner company since it is commonly performed in many different roles, such as test engineers and developers. Test engineers analyse logs as part of their fault localisation workflow when investigating issues produced by automated tests. We worked with a team consisting of five developers developing the tool.

Test engineers at the partner company tend to be highly specialized in specific domain areas of the automotive system, since each module is handled by different teams and requires a certain expertise to troubleshoot effectively. Practitioners rely heavily on their knowledge and past experiences with similar issues when they are localising a fault.

4.3 Extending the Tool

The LLM-based log analysis tool had been developed prior to the study, by a small team at the partner company. The tool only supported a single log format, Diagnostic Log and Trace (DLT). We added support for an additional log format, One Software Download Body (OSB). We investigated log formats that were most widely used by practitioners at the partner company, and asked them which formats would be most helpful to summarize. Both log formats are commonly used to diagnose issues during software installation of the automotive vehicles at the partner company. DLT logs are an open standard, while OSB is a proprietary format. The implemented log formats, DLT and OSB, can be categorized as platform and proprietary logs, respectively [22].

Before the primary data collection, we gathered opinions on the most sought-after changes from early users, to inform potential improvements for the tool. The purpose for adding OSB was two-fold: purpose of gaining familiarity with the tool and expanding the possible use cases for it. Unfortunately, due to the study being designed such that participants themselves selected the issues, only one of the selected issues were of this type.

To extend the tool with new log formats we had to modify several parts of the system, such as the pre-processing and chunking steps, and the two LLM prompts in the reduce and map steps. The system also required some internal modifications to certain implementation details that vary between log formats. The pre-processing step is dependent on the format of the log type, and is likely to be different for every log type. For example, some logs may have timestamps that should be removed, while other formats may not contain this information. The rules for determining start and end of chunks is modified on a per log type basis.

Since the additional format, OSB, is similar to the already implemented DLT format, changes to the prompt templates were minor. The example of the format in the prompt should reflect the log type that is being processed, and the instructions should also be relevant to the contents of the log type.

4.4 Population and Sampling

The study was carried out in individual sessions with each participant, consisting of the main data gathering activities: a think-aloud portion followed by an assessment of cognitive load, and finally semi-structured interview. Half of the sessions were conducted with the help of the LLM-based log analysis tool, while the other half

Table 4.2: Table of study participants. Number of years worked at the company and prior experience with fault localization can be seen, also how often they analyse logs or do fault localization in their daily work. Integration level is the four levels of testing granularity the parter company utilises, described in Section 2.1.1. The last two columns shows which participants were paired together for the same issue and if they analysed that tool manually or with the help of AI-generated summaries.

ID	Years at the company	Years of experience	Integration level	Fault(s)	Treatment
1	9	13	All levels	1	AI
				2	Manual
2	3.5	18	All levels	1	Manual
				2	AI
3	3.5	3.5	Component / Domain	3	AI
4	2.5	1.5	Component / Domain	3	Manual
5	6.5	5.5	Domain	4	AI
6	3+	3+	Domain	4	Manual
7	1	3	Complete	5	AI
8	3	3	Complete	5	Manual

were conducted without the use of the tool, reflecting a traditional fault localisation workflow. We included an additional set of questions before the think-aloud activity in the sessions using the tool, focusing on the participants expectations and prior experiences with LLM-based tools.

Participants were selected using convenience sampling based on their availability, and domain area expertise related to the faults that needed to be analysed. Participants were selected in pairs, in which both participants are specialized in the same domain area. Before the sessions, all participants were asked demographic questions, which can be found in Appendix B.1. All our participants work with fault localization daily. A compiled table of all the answers and participants can be found in Table 4.2. Sessions are carried out individually for one participant at a time. It should be noted that participants that were experienced in the same domain area tended to be colleagues working in the same or adjacent teams. They were instructed not to speak to each other about anything that happened in the sessions or about the fault.

4.5 Data Collection

The troubleshooters that volunteered to participate in the study were set up to do a think-aloud session were they were tasked with localising a fault, with or without the LLM-based log analysis tool. After completing their task, or running out of time, they answer a set of questions to assess their cognitive load, using NASA-TLX. The

Figure 4.3: Each fault is analysed by two test engineers who are experienced within the domain area of the fault. The fault is analysed once by each person: once with the tool and another time without.

sessions then conclude with a semi-structured interview based on their experience with the tool.

Due to the specialised knowledge required to debug specific domain areas, it is difficult to find more than two practitioners that would be able to effectively localize the same fault. The low availability of suitable participants restricted the kind of study that was possible to conduct with our limited time and resources. To address the limited number of participants, we opted to assign one issue per pair of study participants. The pairs were chosen to be as similar in area of expertise as possible.

4.5.1 Think-Aloud Sessions

The think-aloud sessions consisted of the participant attempting to localize the fault for the chosen issue, and explaining their thought process to the researcher. Participants were simply asked to carry out their workflow as they normally would when localising a fault. Those who attended a session with AI-generated summaries, the summaries for the logs related to the issue they were investigating was generated before the sessions and provided to the participant by us. The complete think-aloud session protocol that was used to set up the sessions can be found in Appendix A.

Figure 4.3 illustrates how a pair of participants is set up to analyse a given issue. Each pair of participants was tasked with analysing the same issue but in different sessions: one participant using the LLM-based log analysis tool and the other participant analyses the logs as they would normally carry out their work. The session as a whole lasted 2 hours at most, often shorter due to the participant finishing the think-aloud portion earlier. The think-aloud activity was capped to a maximum of 90 minutes, to give participants equal opportunity to analyse the issue. If the time were to run out, the participant would stop analysing the issue and leave their progress as is.

Before booking a session, the participants were instructed to contact us whenever they had received an actual issue from the system under test, i.e., before they started to analyse or look at it. We decided to avoid pre-selecting the issues ourselves to avoid selection bias of issues and provide the participants with a more realistic experience that would mimic their regular work. The main criteria for selecting an issue was that: it should have an associated log file that is supported by the tool, the participant should have very limited exposure to the issue before the session, the issue should be representative of an issue they would normally analyse. Thus,

Table 4.3: Issues that were analysed by participants in the study, in addition to the type and size of the log files.

ID	Log Type	Log Size
Issue 1	DLT	16.4 MB
	OSB	47.7 MB
Issue 2	DLT	46.9 MB
Issue 3	DLT	7.68 MB
Issue 4	DLT	489 MB
Issue 5	DLT	726.9 MB

the participants were tasked to bring an issue themselves, which they would have normally worked on themselves. A compiled table of all the issues analysed in the case study can be found in Table 4.3.

4.5.2 NASA-TLX

NASA-TLX was used to measure cognitive load of the study participants — directly after the think-aloud sessions have concluded, during which they use the LLM-based log analysis tool. The NASA-TLX score consists of six dimensions, presented in Table 2.1. One of the dimensions, physical demand, was omitted for this study since it is not applicable for the tasks that are carried out. Measuring NASA-TLX was done using two questions, which was then combined to produce the final score. The participant was presented with the explanations of each dimension, shown in Table 2.1, before answering the questions. They also had access to viewing the table on their own while answering the questions.

First, we asked the participant to weigh each dimension against the others in pairs. The participant chooses the dimension they believe to have contributed most to their perceived cognitive load. After a total of 15 comparisons have been made, the number of times each dimension was picked over another one is counted. For instance, a dimension can be chosen 4 times at most, which means that the dimension is weighed more heavily than any other dimension, and is given a weight of 4. A weight of 0 means that the dimension is not relevant to the participant.

The second question asks the participant to rate each of the five dimensions separately, on a scale from 0 to 100 in five-step intervals. To produce the combined augmented score, each dimension from the second question is weighted by its importance (the number of times it was chosen in the pairwise comparisons of the first question). Finally, the augmented scores are summed and the total of them are divided by 10, the number of combinations the pairwise comparison encompassed. This is the final NASA-TLX score for the workload of the task.

4.5.3 Semi-Structured Interviews

Semi-structured interviews conducted during the sessions served as the primary source of qualitative data used to assess participants' satisfaction with the LLM-based log analysis tool. These interviews were designed to gather in-depth insights into the users' experiences, perceptions, and opinions regarding the tool's functionality, usability, and overall impact on their workflow. The flexible nature of semi-structured interviews allowed us to probe further into specific topics that emerged during the conversation and think-aloud session, enabling a better understanding of user thoughts and opinions.

The interview instrument used to guide the interviews was developed to address key areas aligned with the research questions, including perceptions of the log summaries, their workflow analysing logs and AI-tools in the work. The complete interview instrument can be found in Appendix B.

The interview and think-aloud sessions was recorded and auto-transcribed using Microsoft Teams[1]. The researchers also took notes of citations they thought was extra important or interesting.

4.6 Data Analysis

The data collected during the study was analysed using a combination of quantitative and qualitative methods to gain comprehensive insights into the impact of the LLM-based log analysis tool on satisfaction level, cognitive load and workflows. The results from NASA-TLX was plotted and analysed to answer RQ1 regarding cognitive load. Themes from the think-aloud sessions and a thematic analysis on the interviews was used to answer RQ2 and RQ3 regarding satisfaction level and impact on existing workflows.

4.6.1 Data from NASA-TLX

To facilitate an in-depth analysis of the cognitive load data, we calculated the mean, median and standard deviation, as well as generated various plots for each dimension and the total calculated score. The plots included box plots and heat maps that illustrated the distribution and variation of workload scores across different participants and dimensions measured. The box plots provided a clear overview of the median, interquartile range, and potential outliers, while the heat maps visualized the intensity of workload scores, enabling quick identification of patterns or significant differences between the sessions with and without AI assistance. These plots helped us detect trends, outliers, and the overall impact of the LLM-based log analysis tool on the participants' perceived cognitive load.

[1] `https://www.microsoft.com/microsoft-teams/`

4.6.2 Thematic Analysis

Thematic analysis [8] was performed based on conversations with the study participants during the semi-structured interviews, after the think-aloud sessions. The think-aloud activities produced insights on the fault localisation workflow of practitioners at the partner company, which was discussed in further detail in the semi-structured interviews afterwards. We have chosen to conduct thematic analysis on the interviews rather than the think-aloud sessions, due to the interviews being shorter and containing more nuanced commentary.

Each interview had a code extraction and qualitative analysis performed. First, citations from the interview transcripts were coded with codes explaining their relevance for the research questions, using Taguette[2]. Then the codes were divided into themes and sub-themes.

To be able to divide the work of coding the interview transcripts, we had to agree on what codes were relevant for our study and research questions. To do this, first we agreed on one interview to make a thematic analysis independently. Each of us coded what we thought was relevant in Taguette. Then we calculated the total amount of codes and the total amount of codes that we agreed upon. This resulted in a score of 0.77, which was considered sufficient to continue the thematic analysis independently.

[2] `https://www.taguette.org/`

5

Results

This chapter presents our findings from conducting the cognitive load assessments and thematic analysis. Following the results for cognitive load, we present the themes that were derived from the semi-structured interviews.

5.1 Cognitive Load (RQ1)

The cognitive load of the study participant was assessed directly after the think-aloud portion of each session, based on the NASA Task Load Index framework. The results reflect how the tool and log analysis task performed in the session was perceived by the participants in terms of the NASA-TLX dimensions. The results show measured values for the same issues both using the tool and without using it, since each issue was analysed by participants for both methods. Recall that each dimension scale is 0–100, but participants also assign weights to each dimension such that the specific values can be above 100.

The mean, median, and standard deviation of the total NASA-TLX score is shown in Table 5.1, separated for sessions using the tool and without using the tool. The scores are also grouped by dimension. The total NASA-TLX score should be interpreted as the combined value of each dimension for the task performed. A higher total score indicates that participants experienced the session to be more cognitively demanding, while a lower value means the opposite.

The total NASA Task Load Index score in Figure 5.1 indicates that participants using the LLM-based log analysis tool experienced higher cognitive load than without using the tool, but the difference between the two groups is not significant enough to draw any strong conclusions, considering the rather low sample size of 10.

Interestingly, there is a higher deviation of values in sessions where the tool was used, as opposed to the manual sessions. While this could be attributed to the low sample size, we hypothesize that participants had more varied experiences with the tool, or were unsure how to rate their perception of it. In fact, several participants expressed a certain level of confusion about the metric itself, indicating it may have been difficult for them to assess their cognitive load accurately.

Before combining the dimensions into the total score, each dimension is weighted

Table 5.1: The mean, standard deviation and median for total NASA-TLX and unweighted scores of every dimension measured in NASA Task Load Index, grouped by sessions using the tool (AI) and without using the tool (Manual), and rounded to the nearest two-decimal number.

	Total	**Mental**	**Temporal**	**Performance**	**Effort**	**Frustration**
Mean:						
AI	66.70 ± 7.40	79 ± 9.62	43 ± 18.57	62 ± 21.97	65 ± 15.41	37 ± 27.97
Manual	56.7 ± 12.50	56 ± 28.59	38 ± 29.50	68 ± 14.83	42 ± 21.39	25 ± 18.03
Median:						
AI	63	80	40	70	65	20
Manual	58.50	70	30	70	30	30

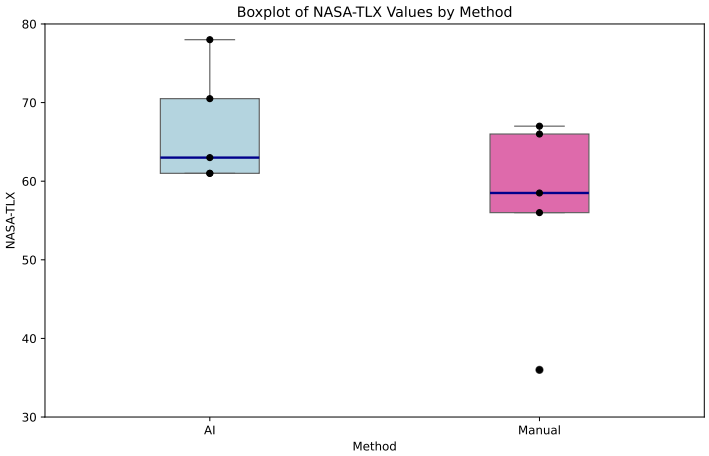


Figure 5.1: Boxplot of the total weighted NASA Task Load Index score for all 10 samples. Value for sessions using the tool in blue to the left, without using the tool in pink to the right. Indicates that participants experienced higher cognitive load when using the tool.

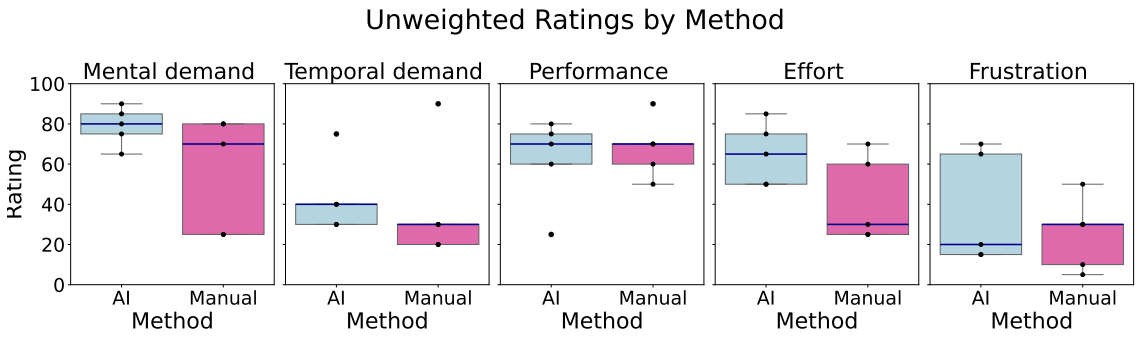


Figure 5.2: Box plots of unweighted values for each dimension of NASA-TLX.

based on the pairwise comparisons, where the participant assessed the dimension to contribute less or more to their overall impression. For example, if a participant rates mental demand as high, but considers it unimportant (i.e., low weight) in the pairwise comparisons, this dimension will contribute less to the total score than others. We show the unweighted values in Figure 5.2, to highlight the differences between methods throughout the dimensions. Note that we see larger differences for mental demand, effort, and frustration. Therefore, even though the total NASA-TLX score includes the weights, it can be of interest to examine the unweighted values as well. In this case, we can see that the participants experienced rather substantially higher mental demand when using the tool.

The LLM-based log analysis tool exhibited much less variation than the manual method in the dimension of mental demand, with participants on average showing slightly higher values using the tool. Frustration was on average higher using the tool. Again, in relation to the sample size, the difference between methods in these dimensions is not significant enough to generalise any conclusions based solely on the measured values of NASA-TLX. Nonetheless, the relative differences between with/without the LLM-based summary do not indicate a clear improvement in the cognitive load of log analysis, as can also be seen in Figure 5.3.

5.2 Thematic Analysis (RQ2, RQ3)

This section presents the themes and sub-themes that were generated through thematic analysis of the semi-structured interviews, as well as some statements from the interviews that were coded for that sub-theme. The analysis produced 5 main themes with a total of 17 sub-themes, as depicted in Figure 5.4. The identified themes can be found with an explanatory description in Table 5.2 and their corresponding sub-themes, also with explanatory descriptions, in appendix E.

Insights gained during the think-aloud portions were often mirrored in following interview. In turn, the interviews were useful as we could go into greater detail during the semi-structured interviews and ask more questions without distracting from their fault localisation process. Even though the workflow activities differed between individuals and teams, participants expressed many of the same opinions

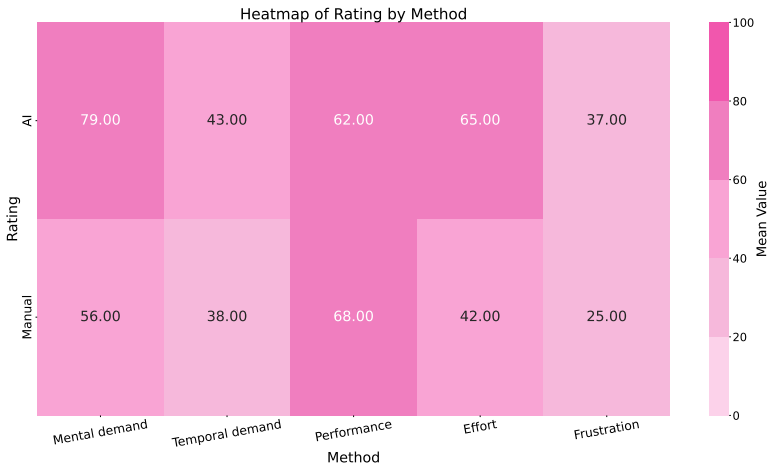


Figure 5.3: Heat map of the mean value of the unweighted rating scores for each dimension measured using NASA-TLX, indicated by the participant for each session, grouped by sessions done manually and with AI-generated summaries.

Table 5.2: The five main themes identified in the thematic analysis. Each theme is complemented with a short description.

Theme	**Description**
Hindrances for adoption	What can hinder the tool from being implemented today
User observations	What the participants initial thoughts of the tool is
Improving the tool	Aspects of the tool which did not live up to expectations
Troubleshoot guidance	How the tool can improve troubleshoot guidance
Improving the workflow	How the current workflow could be improved

related to their impressions of the tool. Some general observations about the process follows in the next section.

5.2.1 General Observations

Most of the study participants decided to look at summary only during the start of the session. Once they had begun to investigate the logs themselves, they tended not to return to the summary. Some study participants forgot to look at the pre-generated summaries for the logs, before being reminded by the researchers. Several study participants decided to first look at the log itself, before reading the summary.

Some study participants were confused as to the contents and inputs of the summaries. This was expressed when participants asked us about which prompts were used, and whether the fault report was used in the generation of the summary.

Several study participants were using a set of predefined filters that was shared in their teams. However, each team had different filters. The filters consisted of certain keywords that were commonly search for, these were saved to a file which they could load into their log viewing program. The LLM-based log analysis tool also had some basic settings for filtering the logs by certain criteria, such as choosing a set of application identifiers that would be whitelisted. However, summaries generated by us prior to the session did not use any settings. The logs pertaining the issue were sent to us in advance of the session, so that we could pre-generate the summaries and save some time during the actual session. However, some study chose participants generated new summaries during the session, by adding certain apps they found out were relevant to the issue. All participants who generated new summaries this way found those to be significantly more useful than the non-filtered summaries. Every participant was given the option to do so, however not all decided to take the opportunity, and only made use of the summaries that were generated by us.

5.2.2 Hindrances for Adoption

This theme indicated that there is potential for making the tool easier to adopt for different types of users, mainly newcomers as opposed to experienced troubleshooters. Participants agreed that the tool may mislead inexperienced users, partly due to the summaries including information not relevant to the issue. However, it did not noticeably hurt the productivity of participants during the study.

Several participants agreed that the summaries that were generated by the tool **could end up misleading users** if they do not have enough experience with troubleshooting, or the tool itself. Participants gave two reasons for this conclusion: the tool does not omit irrelevant information, and the tool is not able to assess the importance of log messages. It is evident from these statements that the participants possess experience and domain knowledge that the LLM does not.

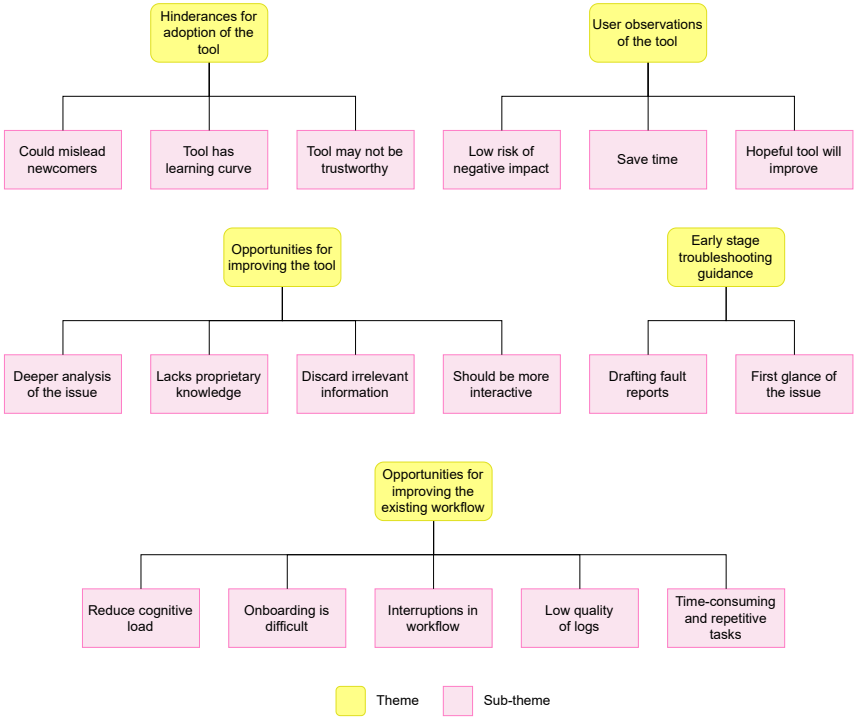


Figure 5.4: The 17 sub-themes are grouped into 5 overarching themes. Connections are shown as lines between themes (rounded) and sub-themes (squared).

> "The tool can draw conclusions that are not as important as in reality. I saw that it drew conclusions that said like "this is bad" or said "this is not correct", I recognize that as something that is normal behaviour, but if I wasn't aware of that, maybe I would be mislead."
>
> - Participant 1

Several participants indicated that **the tool has a learning curve**, i.e., it may take some time and consistent usage before the user learns to effectively integrate the tool into their daily workflow, and to develop an understanding of what they should expect of the summaries. We believe there is a learning curve in gaining an intuition for what types of errors the tool tends to pick up on, and which errors it usually misses. By developing this intuition the user would gain more confidence in the conclusions they draw from reading the summaries.

> "Over time, you would know what you can expect of the tool absolutely to trust it. Yeah, it's that you really know how it works and understand the results of it and then it becomes more like a dialogue instead of someone just telling me this is what I know, then you can recognize this? I mean, hopefully it will be a tool that I can use."
>
> - Participant 1

A few participants indicated concern that the LLM-based log analysis tool may produce summaries that **may not be trustworthy** to be correct and accurate. This was mostly because of the irrelevant information presented that didn't help the participants specific issue they analysed. Some participants expressed worry that the tool might miss some crucial details or include misleading recommendations.

This could potentially lead to incorrect conclusions about the log analysis.

> "I can't be sure if what I'm seeing is true or not."
>
> - Participant 1

> "And I think it would help, if it was accurate enough that we could trust it."
>
> - Participant 4

5.2.3 User Observations

This theme involves the participants' initial thoughts and attitude towards the tool. The participants generally agreed that the LLM-based log analysis tool presents a low risk of negative impact on their workflow if implemented. Most participants expressed that, even if the summaries were imperfect, the overall impact on their work was positive because the tool could help focus their attention and provide a quick preliminary understanding of issues, ultimately leading to better productivity with minimal risk.

Most participants agreed that the LLM-based log analysis tool have **little or no risk to have a negative impact** if it would be implemented. At worst, the summary is not relevant for the issue that is being analysed. Since reading through the summary and coming to that conclusion would take reasonably quick, it was viewed as a neutral impact. At best, the summary perfectly identifies and explains the fault for the troubleshooter, which in that case no further analysation or reading into logs is needed.

> "If it does good, then I know what I should look for and be quicker at pinpointing it in the log. If it doesn't, yeah, then I need to do my job anyway. So I mean, I don't lose that much, I only see that I gain stuff."
>
> - Participant 4

> "So I guess it's more of a win win. I mean, either you still have to look at it manually or you get a good boost and save a lot of time and know where to start to look in the issue as well."
>
> - Participant 3

Most participants indicated that the LLM-based log analysis tool have **great chance to reduce analysation time** in the long run. A few participants who analyses multiple small issues every day said that since they already analyses small issues, the tool might not save too much time on each issue. But even saving $\tilde{5}$ minutes on a issue that would take them 15-20 minutes to analyse manually, can make a big difference on a day with many of those issues.

> "Maybe I would have spent more on analysing the logs, but then since we have multiple test cases each day, I mean at the end of the day it kind of adds up."
>
> - Participant 3

Most participants are positive and **hopeful that the tool will improve** with time and have possibility to be used in large scale. Several participants emphasized that the current limitations, such as the summaries including information not applicable for the specific issue (even if it was included in the logs) and the lack of domain-specific context, hinder its effectiveness.

> "I mean not 100% sure it needs to be AI generated, but at least a tool that can facilitate the analysis of the logs would be beneficial absolutely."
> "I mean, it's all a matter of how how you train the model and and the inputs to the model, but I'm having a great open mindset to that and I hope that it can be useful."
> - Participant 2

> "And I think it would help, or it would help, if it was accurate enough that we could trust it."
> - Participant 4

5.2.4 Improving the Tool

Participants unanimously agreed that context and domain area knowledge is important to draw meaningful conclusions about the logs.

Most of the participants that used the LLM-based log analysis tool agreed that the tool need to make a **deeper analysis of the issue** to be able to give any relevant output that the troubleshooter can make use of.

> "So I guess the first one did not really provide me with any related information to the issue"
> - Participant 3

> "Because I didn't get the relevant information here, not that matches the description of the problem."
> - Participant 2

There were some concern raised by several participants that the tool **lacks proprietary knowledge** and details on the codebase, documentation, tests, etc. from the partner company. This can lead to the tool not being able to make conclusions accurately.

> "What was it missing? Preconditions, I think. Like a simple understanding of what a test is supposed to do."
> - Participant 1

> "Because right now it's just random log messages and it's probably not trained with any automotive logging previously. It's probably trained with a lot of TCP/IP, Linux networking logs. Like if you search for log analysis tools on the internet, it's almost always DevOps like some kind of server."
> - Participant 1

The participants unanimously agreed that the tool **includes irrelevant informa-**

tion in the summary that is not related to the fault or issue that is being analysed. For example, the logs could include messages from an app or fault that is not part of the issue the participant analyses. The tool might not know this so it makes a summary and recommendation for this, which is irrelevant for the participant, so they ignore it.

> "And cause, it's very agnostic. It assumes that everything that it sees is more or less equal to everything else."
>
> - Participant 1

A majority of the participants agreed that the tool **should be more interactive**. Either by enabling the user adding more context before the summary is made or by making it an iterative process. The iterative process could either be a chatbot-solution or that the troubleshooter can add more to the prompt after the first summary is made, to then create another summary.

> "If I were able to set up my own prompt as a precondition? Yeah. Like first you have automatic summary, and if I think like maybe this is towards this area to go into the tool, edit the prompt, run it again and see do I get a better result or not and give it a few attempts?"
>
> - Participant 1

> "if there's like an interface where I could say at this time range we disconnect tester and then it finds and confirms: *yeah that occurred, that's here.* That would be nice."
>
> - Participant 5

5.2.5 Troubleshoot Guidance

The participants agreed that the tool may improve early stage troubleshooting guidance, reducing time from when a participant starts to analyse to when a fault report is created. This initial guidance can serve as a valuable starting point, then the practitioner could start working on a fault report quicker, to continue working in it while the analysation continues.

Several participants said that the tool can help them create and **draft a fault report** quicker.

> "If I would see a fatal log like this and the summary says it's a fatal log, I'll probably start to write it in 10 minutes instead of later. Maybe I wouldn't submit it immediately, but I would still have a draft."
>
> - Participant 1

Most participants said that the tool could give them a **first glance of the issue**, or an indication of where or what to look after. This is in contrast to if the tool could give them a deeper understanding of the issue.

> "It gave a good indication to what could be the issue."
>
> - Participant 3

5.2.6 Improving the Workflow

Participants highlighted activities in their existing workflow that could be improved. The participants agree that the tool should reduce the cognitive load, as the manual process quickly ends up becoming very complex and comprehensive as logs get larger. Some participants also had an idea that the tool should improve the onboarding process, as today there is a lack of documentation on it.

Several participants explain that there is very much information to handle in the manual analysis, and long and complex logs takes a lot of **cognitive load which could be reduced** by the LLM-based log analysis tool

> "I think it's partly like information overload. There's so much logging. I have like thousands of rows, and since we don't have the complete system, we have parts of the system rest is simulated or missing and get a lot of errors that are expected."
>
> - Participant 8

> It would be good to have a quite short summary, at least of what the what the tool thinks that the problem is, and also highlights some interesting lines in the log that the tool has found to use when it attempted at the conclusion.
>
> - Participant 2

Several participants explained how generally **onboarding is difficult** and the hand over for new troubleshooters is very hard. There is not much documentation and most knowledge comes from hands-on experience.

> "So what I was saying is the learning curve for the newcomers is just expanding more and more because we are working with many different things. And it becomes time consuming so it's mostly hands-on practice. So that was most difficult."
>
> - Participant 6

> "I think it's, I don't know. It might be unfeasible because a system like this complexes like this. It changes several times per year and then you have to keep everything updated. It's probably very hard. But I assume maybe the developers have some internal documentation for what they do, but I mean to propagate it, that's up to for someone like me to understand. That's someone that doesn't have the same set of skills to understand the system. I think it would be hard."
>
> - Participant 8

Almost all participants explained that they need to make several **interruptions in their workflow** when analysing logs. The most common interruption is the need to ask colleagues explanatory questions. This can involve asking the developers about the code or other troubleshooters if they have seen a similar issue.

> "It's a flawed system like in that sense. Sometimes there's no other help than them to get someone to help you."
>
> - Participant 8

> "I need to contact different teams, so I need to contact the team that works with the shuttle management and also the reporter and ask him about his rig."
>
> - Participant 2

> "You need to involve other parties as well"
>
> - Participant 3

A good majority of the participants raised complaints about lack of consistency of the logging and that developers **did not adhere to the set up logging standards** that was agreed upon.

> "All applications have their own way of writing the logs."
>
> "...also, the severity levels. It's not uncommon that fatal logs is not fatal."
>
> "If people would actually follow the standard we have set. There is a requirement on how to write logs. But they won't do it. Sadly, it's not reinforced."
>
> - Participant 1

There were several participants who raised the tasks of troubleshooting can be **time-consuming and repetitive**, something that can interrupt their feedback loops and motivation for the task.

> "I mean but downloading logs and manually opening them, enabling filters and searching stuff. I mean, that's something that could ideally be automate those automated. Yeah. So, so those are rather repetitive that there are some steps you do each time for every? Yeah, yeah, definitely. Especially if you have maybe several retries and stuff you have to do it several times and load all the filters again."
>
> - Participant 8

6

Discussion

In this chapter, we discuss and summarize our findings, aiming to provide meaningful insights to practitioners in the industry, and also to inspire future studies to look further into the subjects we have explored in this thesis. The proposed research questions are answered one at a time by each following section.

6.1 Effect on Cognitive Load (RQ1)

The results suggest that the cognitive load of participants was higher when AI-generated summaries were added to the workflow, contrary to what was initially hypothesized. The mean and median values of NASA-TLX were greater for sessions with summaries, compared to manual sessions. The box plot in Figure 5.1 also indicate a slightly higher cognitive load for sessions with the summaries in comparison to those without. It should be considered that the increased cognitive load may be a result of the participants having to understand and interact with a new tool, as opposed to just doing what they are used to doing every day. The tool may not have had enough of a positive impact to offset the increased effort required to change their current workflow. However, we also argue that the increased cognitive load may be caused partly by the lack of issue-related context available to the tool. If the tool cannot make a proper decision of what is relevant or not for the specified issue based on the logs alone, the summaries can confuse or negatively impact productivity of the practitioner.

> **Finding 1:** Results indicate that introducing an LLM-based log analysis tool into a manual troubleshooting workflow can increase the cognitive load of users, which we argue may be connected to the most commonly raised concern of the tool: lacking context and information related to the issue.
> **Recommendation 1:** Design the tool so that it has access to relevant context about the issue, potentially in the form of a fault report or test case description. This information can simply be included in the prompt to the LLM.

Furthermore, the unweighted scores for the dimensions Temporal Demand and Performance does not seem to be affected much by usage of the generated summaries, as can be seen in 5.3. However, in Figure 5.2 we observed a larger spread for Frustration, indicating that certain users were frustrated with the tool. We attribute the higher frustration and cognitive load to the numerous messages mentioned in

the summary, which were often irrelevant to the issue. Participants were sometimes confused by the suggestions in the summary, and had to spend additional time and energy to interpret and assess whether the information in the summary is useful. A commonly used feature of other programs which the participants used was keyword filtering, and pre-saved filters.

Finding 2: The results suggest that the participants experienced higher Mental Demand, and sometimes more frustration when using the generated summaries. Participants appeared to be confused by extraneous information in the summaries, requiring increased effort to parse and understand.

Recommendation 2: Implement more sophisticated pre-processing for log messages, before handing over to the LLM. Additionally, intelligent use of context related to the issue may facilitate automatic keyword filtering.

6.2 Satisfaction Level (RQ2)

The impact of using automatically generated summaries was not as pronounced as we initially hypothesized. Study participants were generally positive towards the tool, but the summaries tended not to live up to their expectations. Many participants found the summaries to be superficial and lacking in-depth information, providing only a high-level overview without elucidating the underlying causes of issues. Other participants reported that the summaries often contained irrelevant or vague information, which made it difficult to derive actionable insights.

The desire to have better control over the resulting summaries was a common theme among study participants. Participants appreciated the initial summaries that were provided, however, some participants wanted to generate new summaries at a later stage of the process when they had a better understanding of the issue. Participants expressed that they would have liked to ask certain questions about the summary to get more specific information, or apply filters to the input to facilitate a more focused summary. For instance, filter based on the error description or specific identifiers they knew were relevant. It was suggested to provide the choice of interacting with the tool similarly to a chatbot. This would enable added the user to add additional context to the input of the tool, perhaps leading to a more useful summary.

Finding 3: Participants requested more interactivity with the tool, and to be able to work iteratively by providing more context of the issue and logs they are analysing.

Recommendation 3: Design the tool so that users can influence the structure and content of the summaries. Additionally, allow for users to use the tool iteratively to better aid them throughout a larger portion of their workflow. For example, design the tool as a chat-bot. Optionally provide additional venues for the user to refine the summaries through prompts or other forms of input.

It was commonly noted that the LLM does not have enough domain knowledge to give a meaningful response to the problem. Proprietary information and specific information about the codebase and its associated artifacts, is not present in the

training data of the LLM. To be more helpful, the tool would also need context about the specific issue that the user is trying to solve, such as having access to the fault report, tests, the codebase, or other artifacts. Several study participants noted that it is much easier to recognize known issues which they already have seen before, as opposed to entirely new problems. Therefore, we argue that it would be useful for the LLM to access documents of active fault reports, as well as historical issues.

Finding 4: The LLM lacks sufficient domain knowledge to provide meaningful recommendations, as it does not have access to requirements, codebase details, or historical fault reports. Improved domain knowledge can help the tool with recognizing known issues and providing appropriate guidance. However, it is still to be investigated if this also helps for new or especially complex issues.

Recommendation 4: Enhance the tool by providing the LLM with access to relevant domain-specific knowledge, such as documentation, historical fault reports, and other proprietary information. For example, this could be accomplished by implementing Retrieval Augmented Generation (RAG). Such capability can help the tool to deliver more insightful recommendations tailored to the specific domain of the issue.

6.3 Improving the Workflow (RQ3)

The participants regularly commented on the depth of the analysis the tool makes. Most agreed that the summaries produced can be a good first glance into the fault, but that a deeper analysis is currently not available and needs to be performed by a human. We suggest that the troubleshooters refer to the summaries in the beginning of their analysis, as a the first step before looking further into the logs. This way, they can get an overall picture of what they will analyse. If the tool is further developed, perhaps it will be able to make deeper analysis and can be used throughout the fault localisation process.

Finding 5: Generated summaries can be useful to provide a first glance that might inform a deeper analysis of the issue in later stages of the workflow.

Recommendation 5: Integrate the tool into the beginning of the fault localization workflow, by presenting it as a preliminary step in the process rather than an aid throughout the whole analysis. Optionally, integrate the tool with the testing software to automatically provide practitioners with summaries before even downloading the logs.

Practitioners performing fault localisation rely on the developers whose modules they are investigating. Low quality log messages, without any useful information, will result in poorer insights from log analysis. For example, study participants agreed that it is common across the industry to see log messages that are inaccurately categorized as an 'error', when 'warning' or 'info' would have been more suitable. Not only do incorrect severity levels throw off human troubleshooters, it also makes it more difficult for the Large Language Model (LLM) to focus on the important messages in a log. When logs contain accurate severity levels and meaningful, well-structured messages, the LLM can more effectively identify relevant and prioritize

relevant information, and potentially generate more precise and contextually aware summaries.

> **Finding 6:** Log messages lack consistent and suitable severity levels.
>
> **Recommendation 6:** Implement and enforce guidelines for selecting appropriate log levels to aid troubleshooters and facilitate better LLM support.

The SPACE framework for measuring productivity, described in Section 2.4.1, suggests that work should be done with minimal delays or interruptions, which is captured by the dimension of **efficiency and flow**. How people talk and work together is also considered in the **Communication and collaboration** dimension of the framework. From the interviews and think-aloud sessions, our data reveals the need for practitioners to interact and communicate with colleagues. For example, asking a colleague who has developed the relevant module if they recognise the issue and could give further explanations. This reliance on communication often required the study participant to pause their analysis in order contact someone else. Due to the company having offices in several time zones, the troubleshooter may have to wait a long time for an answer. In the meantime, the troubleshooter would have to continue working on another issue. This sort of interruption can negatively impact efficiency and flow, and thus also affect productivity.

> **Finding 7:** Dependency on communicating with various colleagues often interrupted troubleshooters in their fault localization workflow, sometimes implying long waiting times.
>
> **Recommendation 7:** If further explanation is needed for an issue, first consult the tool for answers before moving on to asking a colleague. Better support for interacting with the tool to ask questions may result in less workflow interruptions.

6.4 Threats to Validity

There are many activities in the software testing process that require time-intensive manual labour. Anomaly detection, root cause analysis, fault localisation, and fault mitigation are all activities in software testing that are currently being the subject of LLM-based solutions [41]. Some tools for AI-based fault analysis have attempted to develop an all-in-one assistant that performs several types of analyses, such as BugBlitz-AI [47], while we are limiting our efforts solely on fault localisation. By keeping the initial scope small we are able to provide an in-depth analysis on the subject. However, in this study we are focusing on aiding troubleshooters and test engineers with fault localisation specifically by applying LLM-based automated log analysis.

The study is focused on exploratory research questions, since the subject area still is very new. Therefore, we are focusing on the effects of the proposed LLM-based method in the specific context of the case study instead of investigating how we can improve qualities of the tool such as execution time or accuracy.

6.4.1 Internal Validity

At the partner company, each component of a vehicle requires very specific expertise to analyse. Therefore, every test engineer is specialised in their own domain area, and the fault localisation process may vary significantly between teams. For a given fault, there can at times only be one person who is capable of effectively locating the fault. Due to this constraint, we had to design the case study in such a way that each person is working with their own faults separate from the other study participants. We attempt to mitigate this bias by sampling participants across domain areas, to gather enough evidence for some conclusions in spite of the limitation. Additionally, some problem domain areas may be better suited to LLM-based log analysis than others, which could result in substantially different results between subjects. We make sure to sample subjects evenly from different departments, and present the different levels of expertise transparently in the results.

Due to each pair of participants having different issues, the complexity of the tasks varied between participants. We attempt to mitigate this varying bias between participants by instructing them to select new issues they have not inspected before the study. Since the participants brought the first suitable issue they found — with either of the supported log formats — the issues were representative of a random set of realistic faults at the company.

The study involves introducing a new tool into the workflow of practitioners, which may have a biasing effect on the results of the measurement. The effort required to understand and learn the new tool is likely to increase cognitive load. We considered giving the study participants a presentation of the tool before their sessions, but decided it was simple enough to only warrant a small introduction at the start of the session. Additionally, considering our time constraints we wanted to spend the limited time with our participants on collecting insights. We mainly focused on the summaries produced by the tool, and minimized their interaction with the tool by generating summaries prior to the sessions. The aim of the study is to measure the impact of the tool, and this will inevitably affect the workflow in multiple ways. However, it should be considered that our findings may not accurately reflect the usage of the tool in the long term, accounting for its learning curve. A future study may want to observe the impact of the tool over a longer period of time.

6.4.2 External Validity

All study participants are from the same company, so it is possible that our findings are not applicable to the wider industry. However, we argue that our findings will generalize reasonably well to other companies within the automotive industry. Even though log files are generally lacking a widely used standard format, there is still a substantial overlap between automotive companies. Still, further studies in other parts of the software engineering industry is required to determine whether the same practices are useful at other companies.

The specific context of the case study limits the scope further. We are working

with an LLM-based log analysis tool that is already being developed at the partner company, and we are also tasked with extending the tool with certain capabilities. We only have access to the specific log formats that are used at the company, many of which are proprietary. Naturally, we are not able to disclose certain details of the implementation of the tool.

6.4.3 Conclusion Validity

Due to the limited access to study participants, we may have too few samples for the findings to be statistically significant, as this study has a sample size of 10. The quantitative results are presented in addition to qualitative insights from interviews with study participants. If the statistical conclusions are not correct the study should still provide value in the way of industry insight and will contribute by laying a foundation for further studies through the qualitative results.

6.4.4 Construct Validity

Due to the issues being selected at random, we had little control over the properties of the issues that were selected. In Table 4.3 we can see that most of the issues are quite small (relatively to logs that can be as large as one or several gigabytes (GB). The two largest logs consists of logs as large as 726.9 MB respectively 489 MB. Two medium-sized issues consists of logs between 15-50 MB and the smallest consists of a log that is only 7.68 MB. Since no issue consists of any log larger than one gigabyte, this might have an impact on the results not being an accurate measurement and representation of the population. However, this range of sizes are the most common in the industry, and gigabyte-large logs are less common. Additionally, the variation in log sizes across the selected issues provides a meaningful sample to evaluate the tool's performance, and the consistency observed suggests that the results are stable within the typical log size spectrum. Therefore, while there is a limitation concerning larger logs, the study's conclusions are still supported as applicable to typical industry conditions.

7

Conclusion

This thesis presents research exploring the benefits and disadvantages of aiding manual troubleshooting with a LLM-based log analysis tool. The primary objective was to evaluate whether LLM-based log analysis tools could effectively support practitioners, reduce their cognitive load, and improve overall productivity.

A case study was done at a company within automotive industry. Extending a pre-existing LLM-based log analysis tool, we incorporated support for additional log formats commonly used within the industry, notably proprietary types, to broaden its applicability. The tool was introduced into the workflow of some of their troubleshooters, who manually analyse long and complex logs to localise faults when issues arises in their CI-pipeline. The study employed think-aloud sessions and semi-structured interviews with the participants to collect the data.

The goal of the case study was to examine how a LLM-based log analysis tool would affect the cognitive load and satisfaction level of the troubleshooters, as well as impact the existing manual fault localisation workflows. To do so, we arranged think-aloud sessions and semi-structured interviews with eight troubleshooters from the partner company, who in total analysed ten faults in ten different sessions. We focused on what their cognitive load was for the task, what affected it and their thoughts on the workflow. The results from the sessions was then analysed with a thematic analysis. Our findings indicated that while participants generally viewed the tool positively, the generated summaries was sometimes limited to the context and interactivity given from the troubleshooter.

Our results provide insights on how LLM-based tools can be implemented as an aid into the workflow of troubleshooters, the benefits and limitations of doing so and it's impact on the troubleshooters cognitive load and satisfaction level. Our contributions include a list of recommendations regarding what to consider when providing AI-generated log summaries to ease the work of troubleshooters. The recommendations can be summarised with adding interactivity to the tool so that the user can add context iteratively throughout their analysis.

In this study we have focused on how a LLM-based log analysis tool affects the human practitioners, and how such a tool would fit into their log analysis workflow. Through our findings we recognized multiple potential improvements to the tool,

which could possibly affect the results significantly if these were to be implemented. Unfortunately, we did not have the resources to implement these changes, which is why we think a future study could focus on how such improvements could impact the effectiveness of the tool. Due to the limited sample size of our study, we believe further research on how LLM-based tooling may affect the human in the loop is necessary to gain a better understanding of its benefits and limitations. Additionally, we think it would be beneficial to spend more time developing a more complete tool for the study, potentially evaluating whether recommendations such as the ones in our study would yield more positive results. Finally, studies at different companies, perhaps in other than the automotive industry, would be contribute to more generalizable findings in the future.

Bibliography

[1] Toufique Ahmed, Supriyo Ghosh, Chetan Bansal, Thomas Zimmermann, Xuchao Zhang, and Saravan Rajmohan. Recommending root-cause and mitigation steps for cloud incidents using large language models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1737–1749. IEEE, 2023.

[2] Emil Alégroth, Robert Feldt, and Pirjo Kolström. Maintenance of automated test suites in industry: An empirical study on visual gui testing. *Information and Software Technology*, 73:66–80, 2016.

[3] Mohammad Amin Alipour. Automated fault localization techniques: a survey. *Oregon State University*, 54(3), 2012.

[4] Anunay Amar and Peter C Rigby. Mining historical test logs to predict bugs and localize faults in the test logs. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 140–151. IEEE, 2019.

[5] James H Andrews. Testing using log file analysis: tools, methods, and issues. In *Proceedings 13th IEEE International Conference on Automated Software Engineering (Cat. No. 98EX239)*, pages 157–166. IEEE, 1998.

[6] Rohini Bisht, Selomie Kindu Ejigu, Gregory Gay, and Predrag Filipovikj. Identifying redundancies and gaps across testing levels during verification of automotive software. In *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 131–139. IEEE, 2023.

[7] Barry W. Boehm. Improving software productivity. *Computer*, 20(09):43–57, 1987.

[8] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101, 2006.

[9] Edward Chuah, Shyh-hao Kuo, Paul Hiew, William-Chandra Tjhi, Gary Lee, John Hammond, Marek T Michalewicz, Terence Hung, and James C Browne. Diagnosing the root-causes of failures from cluster log files. In *2010 International Conference on High Performance Computing*, pages 1–10. IEEE, 2010.

[10] Mihaly Csikszentmihalyi and Mihaly Csikzentmihaly. *Flow: The psychology of optimal experience*, volume 1990. Harper & Row New York, 1990.

[11] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[12] Biplob Debnath, Mohiuddin Solaimani, Muhammad Ali Gulzar Gulzar, Nipun Arora, Cristian Lumezanu, Jianwu Xu, Bo Zong, Hui Zhang, Guofei Jiang, and Latifur Khan. Loglens: A real-time log analysis system. In *2018 IEEE 38th international conference on distributed computing systems (ICDCS)*, pages 1052–1062. IEEE, 2018.

[13] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

[14] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 1285–1298, 2017.

[15] Chris Egersdoerfer, Di Zhang, and Dong Dai. Early exploration of using chatgpt for log-based anomaly detection on parallel file systems logs. In *Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing*, pages 315–316, 2023.

[16] Nicole Forsgren, Margaret-Anne Storey, Chandra Maddila, Thomas Zimmermann, Brian Houck, and Jenna Butler. The space of developer productivity: There's more to it than you think. *Queue*, 19(1):20–48, 2021.

[17] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

[18] Shilin He, Pinjia He, Zhuangbin Chen, Tianyi Yang, Yuxin Su, and Michael R Lyu. A survey on automated log analysis for reliability engineering. *ACM computing surveys (CSUR)*, 54(6):1–37, 2021.

[19] Saki Imai. Is github copilot a substitute for human pair-programming? an empirical study. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, pages 123–126, 2022.

[20] Sungmin Kang, Gabin An, and Shin Yoo. A quantitative and qualitative evaluation of llm-based explainable fault localization. *Proceedings of the ACM on Software Engineering*, 1(FSE):1424–1446, 2024.

[21] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural*

information processing systems, 35:22199–22213, 2022.

[22] Łukasz Korzeniowski and Krzysztof Goczyła. Landscape of automated log analysis: A systematic literature review and mapping study. *IEEE Access*, 10:21892–21913, 2022.

[23] Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, and Xuewei Chen. Log clustering based problem identification for online service systems. In *Proceedings of the 38th international conference on software engineering companion*, pages 102–111, 2016.

[24] Yilun Liu, Shimin Tao, Weibin Meng, Jingyu Wang, Wenbing Ma, Yuhang Chen, Yanqing Zhao, Hao Yang, and Yanfei Jiang. Interpretable online log analysis using large language models with prompt strategies. In *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension*, pages 35–46, 2024.

[25] Abi Noda, Margaret-Anne Storey, Nicole Forsgren, and Michaela Greiler. Devex: What actually drives productivity: The developer-centric approach to measuring and improving productivity. *Queue*, 21(2):35–53, May 2023.

[26] Adam Oliner, Archana Ganapathi, and Wei Xu. Advances and challenges in log analysis. *Communications of the ACM*, 55(2):55–61, 2012.

[27] OpenAI. Gpt-4o. https://platform.openai.com/docs/models/gpt-4o. Accessed: 2025-05-22.

[28] Nicole Radziwill and Graham Freeman. Reframing the test pyramid for digitally transformed organizations, 2020.

[29] Md Nakhla Rafi, Dong Jae Kim, Tse-Hsun Chen, and Shaowei Wang. Enhancing fault localization through ordered code analysis with llm agents and self-reflection. *arXiv preprint arXiv:2409.13642*, 2024.

[30] Sarah Rastkar, Gail C Murphy, and Gabriel Murray. Automatic summarization of bug reports. *IEEE Transactions on Software Engineering*, 40(4):366–380, 2014.

[31] Colin Robson. *Real world research*, volume 2. Blackwell Oxford, 2002.

[32] Per Runeson. A survey of unit testing practices. *IEEE software*, 23(4):22–29, 2006.

[33] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14:131–164, 2009.

[34] Max Schäfer et al. An empirical evaluation of using large language models for automated unit test generation. *IEEE Transactions on Software Engineering*,

2023.

[35] M. L. Siddiq et al. Using large language models to generate junit tests: An empirical study. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, 2024.

[36] Software Center. Software center, 2024. Accessed: Dec. 18, 2024. [Online]. Available: `https://www.software-center.se/`.

[37] Daniel Sundmark, Kai Petersen, and Stig Larsson. An exploratory case study of testing in an automotive electrical system release process. In *2011 6th IEEE International Symposium on Industrial and Embedded Systems*, pages 166–175. IEEE, 2011.

[38] Liyan Tang, Zhaoyi Sun, Betina Idnay, Jordan G Nestor, Ali Soroush, Pierre A Elias, Ziyang Xu, Ying Ding, Greg Durrett, Justin F Rousseau, et al. k large language models on medical evidence summarization. *NPJ digital medicine*, 6(1):158, 2023.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[40] Stefan Wagner and Melanie Ruhe. A systematic review of productivity factors in software development. *arXiv preprint arXiv:1801.06475*, 2018.

[41] Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering*, 2024.

[42] James A Whittaker. What is software testing? and why is it so hard? *IEEE software*, 17(1):70–79, 2000.

[43] Ratnadira Widyasari, Jia Wei Ang, Truong Giang Nguyen, Neil Sharma, and David Lo. Demystifying faulty code: Step-by-step reasoning for explainable fault localization. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 568–579. IEEE, 2024.

[44] W. Eric Wong, Vidroha Debroy, Rui Gao, and Yao Li. The dstar method for effective software fault localization. *IEEE Transactions on Reliability*, 63(1):290–308, 2014.

[45] W Eric Wong, Ruizhi Gao, Yihao Li, Rui Abreu, and Franz Wotawa. A survey on software fault localization. *IEEE Transactions on Software Engineering*, 42(8):707–740, 2016.

[46] Chunqiu Steven Xia, Yuxiang Wei, and Lingming Zhang. Automated program repair in the era of large pre-trained language models. In *2023 IEEE/ACM*

45th International Conference on Software Engineering (ICSE), pages 1482–1494. IEEE, 2023.

[47] Yao Yao et al. Bugblitz-ai: An intelligent qa assistant. In *2024 IEEE 15th International Conference on Software Engineering and Service Science (ICSESS)*, pages 57–63. IEEE, 2024.

[48] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57, 2024.

[49] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.

A

Think-Aloud Session Protocol

This is the protocol for the think-aloud sessions performed with each participant.

A.1 Introduction

First the participant is instructed how the think-aloud session will be set up and encouraged to talk as much they can when performing tasks. The participant is then asked for their consent to record audio during the think-aloud session, and are informed that the data that is collected during the session will be anonymized and presented in the study. The session lasts roughly two and a half hours, or shorter if they are done with their tasks sooner. The study participant is informed that it is not important for them to finish the task. A researcher will be present during the session and take notes on what the participant is doing.

A.2 Think-Aloud Steps

Afterwards, they are presented with a sequence of tasks which they are supposed to perform to the best of their abilities within the timespan of 1 hour. When the allotted time is over, they are given a 5-10 minutes for reflecting on the session. The participant is asked to explain their thought process and motivations while performing the tasks.

In case of the LLM-based approach, do step (1) — in case of a manual approach, go directly to step (2).

1. Read the summary for each log. Give thoughts on the quality, accuracy and helpfulness of the summary. Does it provide concrete and actionable suggestions, or is it too vague? Are there any false positives in the summary?
2. Localize the fault. Think-aloud and explain the steps you are taking and why.
   (a) What do you know about the fault so far?
   (b) What is the first thing you'll do when starting to analyse the logs

3. Draft a fault report for the issue. Include as much information as you have been able to gather during the allotted timespan. The fault report will be studied and compared with those of other study participants.

A.3 Interview Questions

To conclude the session, the study participant is asked a set of interview questions to assess their experience with performing the tasks. The questions can be found in Appendix B

B

Interview Instrument

The participant is sent a form with some demographic questions before the think-aloud session and interview. The rest of the interview questions are asked during the semi-structured interview after the think-aloud session. During the interview, the participant is asked questions to assess their cognitive load and satisfaction level. If the participant used generated summaries, they are asked about their expectations on the summaries before the think-aloud session and about their experience afterwards.

B.1 Demographic Questions

The participant is asked a few demographic questions about their experience of working with fault localisation at the company, and their attitude towards LLM-based software tools in general. These questions are answered in a form before the think-aloud session and interview begins.

- How many years have you worked at the company?
- How many years of experience do you have localising faults?
- How often do you analyse logs?
- What test integration level do you work with? (Unit, Component, Domain, Complete)

B.2 Expectations of Summaries

The participant is asked about their expectations and attitude towards AI-generated summaries, in the case that they will be using generated summaries during their think-aloud session. It is useful for the company to know whether their expectations are met so they can continue improving the tool after the research is completed.

- What is your attitude towards AI-assisted tools in your work?
- What is your attitude towards LLM-based log summarisation?
- Have you used LLMs or other LLM-based tools previously?
- Do you think AI-generated log summaries would help you in your daily

workflow?
- What information do you expect the summaries to contain to be able to aid you?
- How would you use the generated summaries as part of your workflow?

B.3 Semi-structured Interview after Think-Aloud Session

The semi-structured interview is conducted directly after the think-aloud session. The questions are divided in the several categories, as outlined in the following subsections.

B.3.1 Questions Related to Cognitive Load (RQ1)

The participant is asked to evaluate their cognitive load during the think-aloud session through a series of questions based on the NASA Task Load Index. The participant does a pair-wise comparison between several dimensions which can be found in Appendix C, based on the dimensions they believe contributed most to their cognitive load. For example, they are asked which was more important for their workload, either mental demand or temporal demand. The participant is also asked to rate each dimension on a scale, which can be found in Appendix D. The participant receives the questions and printed on paper, along with a page containing explanations of the dimensions.

B.3.2 Questions Related to Explaining the Issue

The participant is asked to explain the issue that they analysed during the think-aloud session. Depending on how far they got in the fault localisation process, we ask them to go into further detail. For sessions in which the participant was not able to localize the fault during the allotted time, they are asked to reflect on what information they would have needed to complete their task.

- Were you able to find the fault for the issue?
  - If **yes**, what was it? During which activity did you find it?
  - If **no**, what do you know about the fault so far? What do you need to be able to find it? Do you miss any information?

B.3.3 Questions Related to Satisfaction Level (RQ2)

The following questions were asked to all participants, regardless whether they used AI-generated summaries during the think-aloud session. More specific questions on satisfaction related to the generated summaries and manual fault localisation are specified in the subsequent subsections.

- Did you ever feel frustrated?
- What was the most frustrating moment?

B.3.3.1 Questions for Session without AI-generated Summaries

The following questions were specifically asked to participants that were performing fault localisation manually, without the help of AI-generated summaries. The questions are designed to gather insights on how the existing process can be improved, and how AI-generated summaries could help.

- Did you ever feel stuck and unable to continue the analysation?
  - How did you overcome this?
- What specific steps in the manual process did you find most time consuming?
- What tools or resources did you rely on the most during your manual analysis?
- What does not work about the existing process of analysing logs manually?
  - How do you think it can be improved or aided?
- What was the most difficult activity you did during the analysis?
- Was there an activity where AI-generated log summaries would have helped you in your analysis?
  - Do you think the analysis could be done faster with them?

B.3.3.2 Questions for Session with AI-generated Summaries

The following questions were specifically asked to participants that were given AI-generated summaries during the think-aloud session. The questions are designed to gather insights on how satisfied the participant was with their experience of using the AI-generated summaries.

- Do you feel like this tool made your work any easier? Why/why not?
- How much time do you estimate it would take you to analyse the logs manually? Do you feel like the summaries helped you save time, if so, how much time did you feel like you saved?
- Would it be a harder task if you didn't have the summaries?
- Would you recommend the use of AI-generated summaries to a colleague?
- Would you continue using AI-generated summaries in your daily work if they look like they did today?
  - How would you incorporate them into your workflow?
- Do you think the generated summaries helped you in writing a better fault report?

B.3.4 Questions About Quality of the AI-generated Summaries

The following questions were asked to participants that were given AI-generated summaries during the think-aloud session to assess whether the summaries lived up to their expectations. The questions are designed to gather insights on the perceived quality of the AI-generated summaries. It is useful for the company to know whether their expectations are met so they can continue improving the tool after the research is completed.

- Were the generated summaries helpful?
- Did the generated summaries live up to your expectations?
  - If **yes**, how? What worked best?
  - If **no**, why not?
- Did the summaries provide you with all the information you needed? If not, what was missing?
- How accurate do you think the AI-generated log summaries were?
- Did you ever feel misled by the generated summaries?
  - Why, what and how?

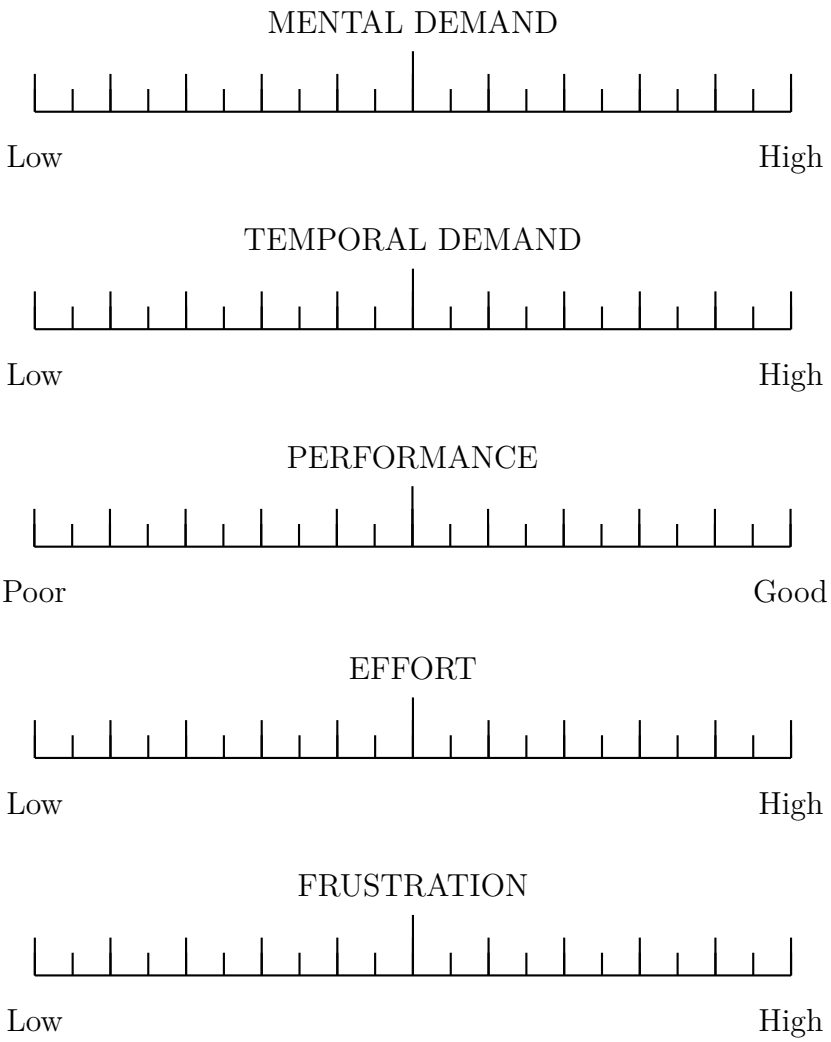
C

Pairwise Comparison

Mental Demand or **Temporal Demand**	**Mental Demand** or **Performance**
Mental Demand or **Effort**	**Mental Demand** or **Frustration**
Temporal Demand or **Performance**	**Temporal Demand** or **Effort**
Temporal Demand or **Frustration**	**Performance** or **Effort**
Performance or **Frustration**	**Effort** or **Frustration**

D

Ratings


MENTAL DEMAND
Low
High
TEMPORAL DEMAND
Low
High
PERFORMANCE
Poor
Good
EFFORT
Low
High
FRUSTRATION
Low
High

E

Sub-themes from Thematic Analysis

Table E.1: The 17 sub-themes identified in the thematic analysis. Related sub-themes are grouped into their respective main themes. Each sub-theme is complemented with a short description.

Theme	Sub-theme	Description
Hindrances for adoption	Could mislead newcomers	Users with little or no experience with troubleshooting might be mislead
	Tool has learning curve	Users might need a lot time to get to know the tool and it's pros and cons
	Tool may not be trustworthy	Users experience mistrust for the tool and are sceptical
User observations	Low risk of negative impact	The tool mainly have positive or neutral consequences of implementation
	Can save time	The tool have potential to save time in the analysis workflow
	Hopeful the tool will improve	Participants are positively hopeful the tool will improve with further development
Improving the tool	Deeper analysis of the issue	The tool's analysis is too superficial to be helpful
	Lacks proprietary knowledge	The tool lacks knowledge of internal language and processes at the partner company
	Discard irrelevant information	The tool does not filter irrelevant information which causes irrelevant outputs
	Should be more interactive	The tool can be improved by being more interactive, e.g. with chatbot-functionality
Troubleshoot guidance	Drafting fault reports	The tool can help with writing fault reports quicker
	First glance of the issue	The tool can give a good indication on where or what the issue might be
Improving the workflow	Reduce cognitive load	The workflow is information heavy
	Onboarding is difficult	The onboarding and hand over for the trouble shooter role is hard to make accessible or
	Interruptions in workflow	Troubleshooters experience they must continuously interrupt their workflow or flow state to continue troubleshooting, e.g., ask colleagues explanatory questions
	Low quality of logs	The logs lacks consistency and adherence to the internal standards set up
	Time-consuming and repetitive tasks	Troubleshooters experience slow feedback loops