From Logs to Lessons: An Exploration of LLM-based Log Summarization for Debugging Automotive Software

Anton Ekström*
Hampus Rhedin Stam*
antoneks@student.chalmers.se
rhedinh@student.chalmers.se
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden

Francisco Gomes de Oliveira Neto
Gregory Gay*
francisco.gomes@cse.gu.se
greg@greggay.com
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden

Sabina Edenlund
sabina.edenlund@volvocars.com
Volvo Cars AB
Gothenburg, Sweden

Abstract

Identifying where faults occur is an essential part of debugging, yet examining extensive system logs can be slow and mentally demanding, especially in complex software environments. One emerging strategy to enhance log analysis is to employ large language models (LLMs) to distill log information into more manageable summaries that can guide human reasoning during diagnosis. We report on a case study carried out in an automotive setting, where engineers investigated actual failures with and without support from an LLM-based summarization tool. During fault localization sessions where participants analyzed real failure logs, we collected cognitive load measurements, observed their reasoning processes, and gathered feedback on both the LLM-based summarization and the workflow through post-session interviews. Our results indicate that although the use of summaries raised certain cognitive demands, particularly related to mental effort and time pressure, participants experienced less frustration overall and considered the support helpful in focusing their attention. They also expressed a clear interest in being able to shape and refine summaries as their understanding evolved. These findings offer insights into how LLM-generated summaries influence practitioners' diagnostic work and point toward the need for more adaptive, interactive, and workflow-aware support.

CCS Concepts

• **Software and its engineering** → **Software testing and debugging**; *Embedded software*; • **Human-centered computing** → *Interaction design.*

Keywords

Automated Software Engineering, Debugging, Large Language Models, Software Logs, Log Analysis

*Both authors contributed equally to this research.

AST'26, Rio de Janeiro, Brazil

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Anton Ekström, Hampus Rhedin Stam, Francisco Gomes de Oliveira Neto, Gregory Gay, and Sabina Edenlund. 2026. From Logs to Lessons: An Exploration of LLM-based Log Summarization for Debugging Automotive Software. In *Proceedings of Automated Software Test (AST'26).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Fault localization is a critical early step in debugging software systems, allowing developers to identify where faults occur before addressing their root causes [33]. In modern software development, this process often involves extensive log analysis, which can be time-consuming and labor-intensive, particularly in large and complex systems such as those in the automotive domain [22, 26]. Another key challenge is the lack of standardized log formats, as many systems rely on proprietary or niche formats [22, 26]. This variability often requires format-specific tools or workarounds, leading to duplicated effort and higher testing costs [3].

Recent advances in large language models (LLMs) have opened new opportunities to support software engineering workflows [19, 31, 32]. First, LLMs are particularly well suited for processing unstructured text. Second, widely used formats may already be represented in their training data, while proprietary formats could be incorporated through in-context learning [13, 24]. Consequently, we argue that LLMs are a promising candidate for assisting in log analysis and summarization. We explore whether LLM-generated log summaries can help practitioners manage the cognitive and practical demands of fault localization tasks in an industrial setting.

We conducted a study in collaboration with a European automotive company, Volvo Cars AB, focusing on the use of an LLM-based log summarization tool to support fault localization. We conducted think-aloud sessions with practitioners, measured cognitive load using the NASA Task Load Index [17], and complemented these results with a thematic analysis of post-session interviews.

Our findings show that while cognitive load did not decrease overall, LLM-assisted sessions yielded lower frustration scores and surfaced clear practitioner expectations regarding tool interactivity. Participants emphasized the potential of such tools to automate repetitive steps and support iterative workflows, aligning with broader goals of improving productivity and user experience in software testing. In summary, the contributions of our paper are:

- An empirical study on LLM-based log summarization in the context of fault localization within the automotive industry.

- Quantitative and qualitative insights into the impact of LLM-generated summaries on cognitive load, user satisfaction, and workflow.
- Recommendations for improving human-LLM interaction in log analysis tools, including interactivity, automation, and integration into existing workflows.

2 Background

Here we explain key concepts related to the instrumentation and evaluation of our approach. The section introduces fault localization, log summarization, and the use of large language models (LLMs) to support these processes.

The aim of the **fault localization** process is to isolate the cause and/or source code location responsible for an observed test failure [1, 33]. This process typically involves inspection of software artifacts including source code, test cases, issue reports, and logs recorded during program execution [33]. These artifacts are used as part of (i) anomaly detection, i.e., identification of abnormal and erroneous program behavior, (ii) localization of code elements that contribute to the failure, and (iii) root cause analysis to explain the underlying reason for the failure [14, 18]. Fault localization can be performed manually, with tool assistance, or—in some cases—fully automated. The fault localization process is typically time-consuming and relies heavily on developers' intuition and experience, leading to great interest in approaches that support or automate aspects of the process [24].

To enhance fault localization and reduce the required manual effort, **log analysis** is often used to provide additional context and evidence during debugging [18, 22]. However, the lack of standardized log formats can make this analysis challenging [22]. Many formats follow similar structures where events or messages are recorded during the execution of the software, generally in chronological order and associated with a timestamp of their occurrence, potentially with associated metadata such as screenshots or other supporting data [3]. Log files may come from a live production environment, but may also be created after execution based on observations gathered during the execution.

Given the volume and level of detail contained in log information, practitioners can benefit from a summarized view that highlights the most relevant elements. **Log summarization** refers to the process of condensing the contents of a log, or a segment of it, to make them more accessible and easier to interpret [15]. Since logs are typically long and require significant effort to examine, automated summarization can help accelerate and simplify analyses [15].

In turn, Large Language Models (**LLMs**) are complex machine learning models suited for language analysis and transformation tasks such as translation, summarization, and decision support [37]. LLMs take as input instructions in the form of a *prompt* [8]. LLMs then iteratively and probabilistically predict the next token—a basic unit of text—to add to the generated output [37]. Because of their ability to analyze and output both natural language and code, LLMs are well-suited for software development tasks including test generation [31], fault localization [19], and program repair [34]. Here, we investigate their ability to perform log summarization.

An LLM and a human may not draw the same meaning from a prompt, which has led to the development of different prompting strategies. Zero-shot prompts contain only instructions on how to produce the expected output, while one-shot or few-shot prompts provide one or more examples of input and expected output [20, 24]. Chain-of-thought (CoT) prompting instructs the LLM to perform step-by-step reasoning steps [20]. In addition to how the model is prompt, the information given in the prompt can also affect the LLM's output. In-context learning refers to the incorporation of additional contextual information into the prompt, such as project-specific details (e.g., coding guidelines, design artifacts, and code dependencies) to guide the model's output more effectively. The ability of an LLM to process prompts and external data is limited by its *context window*—the number of tokens that it can hold in memory [15]. For example, a full software log may exceed this limit, thus introducing the need for summarization [15].

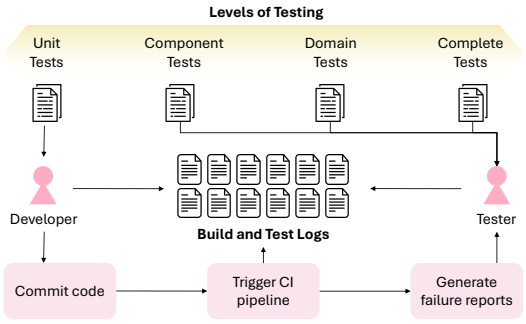

Figure 1: Overview of the log analysis process.

2.1 Case Study Context

The case study was conducted at Volvo Cars AB, a prominent automobile manufacturer headquartered in Europe. We worked with a team consisting of five developers to create and modify an LLM-based log summarization tool.

Similarly to other automotive companies [6, 29], and in line with safety regulations in the automotive industry, Volvo Cars ABfollows a modified V-Model that defines four levels of testing granularity: unit, component, domain, and complete system. Figure 1 provides a simplified overview of how failing tests are handled. Unit test failures are small in scope and can be addressed directly by the developers responsible for the affected modules. However, at higher levels of abstraction, specialized testers with a broader understanding of the system are responsible for analyzing failures. After code is committed to a repository, the Continuous Integration (CI) pipeline executes the test suite. When tests fail, a tester analyzes the log files created during the execution to localize the potential fault, creates a fault report, and assigns the issue to the appropriate team.

Testers at Volvo Cars AB tend to be highly specialized in specific domain areas, since each module is developed by different teams and requires a certain expertise to debug effectively. Practitioners rely heavily on their knowledge and past experiences with similar issues when they are localizing a fault. Consequently, the fault localization process is heavily dependent on log files generated in multiple formats during test execution. These log files can be very large—up to several gigabytes in extreme cases. Currently, developers analyze logs in a largely manual process, using their

```
## 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'.
- 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).
```

Listing 1: Example of a summary that has been generated by the LLM-based log summarization tool.

own experience and intuition, as well as keyword searches, to filter log files and localize failures. Therefore, there is great interest in support for log analysis and summarization.

3 LLM-based Log Summarization

In this study, we have extended and evaluated an LLM-based log summarization tool developed for use at Volvo Cars AB. The tool uses GPT-4o (version 2024-08-01-preview) to summarize data. Figure 2 illustrates the user workflow.

The user begins by uploading a log file to be summarized through a web application. They can then optionally apply filters and customize the instructions for the LLM—e.g., adding a simple description of the error or keywords to filter messages. The user then waits for the tool to finish processing the log file and downloads the summary in Markdown format. The LLM has been instructed to highlight the most relevant messages in the log file, and to provide suggestions on how to further investigate the issue. An example of the summary can be seen in Listing 1.

The tool supports summarization of two log formats in use at the company: Diagnostic Log and Trace (DLT) and One Software Download Body (OSB). Both logs consist of timestamped messages describing the status of the system during execution. DLT is an open standard used in the automotive industry [5], while OSB is a proprietary format used by Volvo Cars AB. Both log formats can be categorized as platform logs [22]. The initial tool supported only DLT, and support for OSB was added as part of this study, as the format is also in wide use across the company.

After receiving the logs, the tool follows the steps illustrated in the highlighted area of Figure 2. Since individual log files can be very large (e.g., hundreds of megabytes to more than a gigabyte), the tool leverages a MapReduce architecture, which applies parallel processing to improve the efficiency of handling large datasets [11].

This architecture structures processing into two stages, called "map" and "reduce". During the "map" phase, a log is broken into smaller "chunks," which are processed by the LLM in parallel. Then, during the "reduce" stage, the summaries of the chunks are merged by the LLM and the final output is returned.

- **Pre-process logs:** Log files are cleaned to reduce their size by removing unnecessary information. Specific characters are stripped, and the logs are segmented so that each segment corresponds to a single logging session.

```
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.
```

Listing 2: 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:
```

Listing 3: Prompt template for the reduce stage.

- **Split data:** Processed logs are divided into chunks based on the available context window. Each chunk contains as many complete segments as possible to optimize the input size.
- **Summarize chunks ("map"):** Each chunk is embedded in a prompt template and sent to a separate LLM instance with a fresh context. All chunks are processed in parallel and in isolation. The prompt instructs the LLM to summarize the key information in each chunk.
- **Combine chunks ("reduce"):** After all chunks have been summarized, the resulting summaries are merged in a final step. The LLM receives all intermediate summaries in a single prompt and produces one consolidated summary.

The prompts used in our approach include a role and a set of instructions to the LLM (but no examples of responses to sample inputs, i.e., zero-shot [21]). We developed the prompts over multiple iterations. The basic prompt template for an individual chunk in the map stage is shown in Listing 2, whereas the prompt template for the reduce stage are shown in Listing 3. The specific prompts vary by log format.

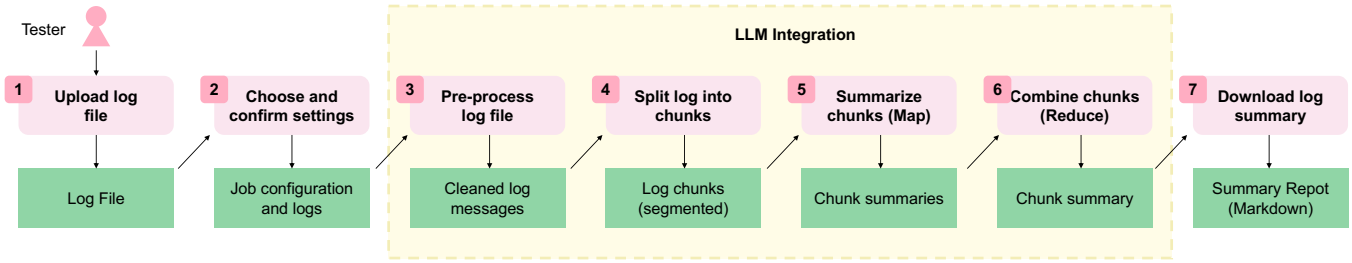

Figure 2: Workflow of a user interacting with the LLM-based log summarization tool. The tasks within the highlighted area are integrated with the LLM used to summarize the logs.

4 Methods

Our primary goal was to explore the impact of LLM-based log summarization on the log analysis workflow at Volvo Cars AB. Our case study is framed using the guidelines by Runeson and Höst [28]. In particular, we address the following research questions:

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

Cognitive load is a major component of developer productivity [25] and, for **RQ1**, we associate it with the ability to parse, learn, and retain information from logs. **RQ2** assesses practitioner satisfaction—another core component of productivity that plays a significant role in tool adoption [16]. Lastly, **RQ3** explores how the practitioners' log analysis workflow may be affected by the LLM-based log summarization tool, and the potential benefits or drawbacks of the use of this tool as part of their workflow.

Figure 3 illustrates the activities conducted to address the research questions. Data was collected through think-aloud sessions in which participants performed fault localization either manually or with support from the LLM-based summarization tool, followed by semi-structured interviews. Cognitive load was measured using the NASA Task Load Index [17]. Practitioner satisfaction and perceived workflow impact were examined through thematic analysis of interview transcripts and notes from the think-aloud sessions.

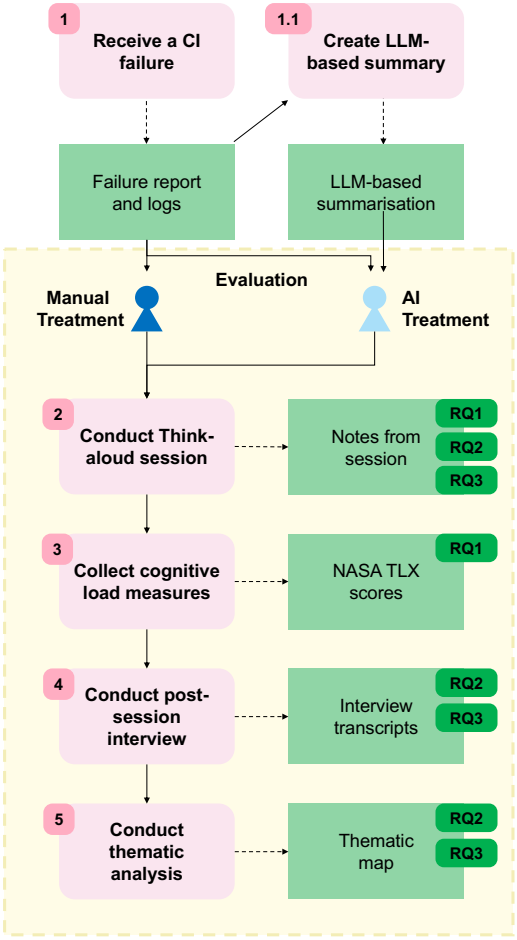

Figure 3: Overview of the performed research activities (rounded rectangles), and corresponding research artifacts (rectangles). We indicate which research artifact is used to answer the different research questions.

4.1 Participant Sampling and Data Collection

Table 1 presents the participants of our study. Participants were selected using convenience sampling based on their availability and domain area expertise relative to the faults to be localized. All participants work with fault localization daily.

Participants were selected in pairs, where both participants were specialized in the same domain area. Sessions were then carried out individually for one participant at a time. Participants that were experienced in the same domain area tended to be work in the same or adjacent teams. Therefore, they were instructed not to speak to each other about the fault.

The study was carried out in individual sessions with each participant. Each session consisted of (i) a think-aloud observation where they were tasked with localizing a real fault (with or without the LLM-based log summarization tool), (ii) assessment of cognitive load based on the NASA Task Load Index, and (iii) a semi-structured interview. Half of the sessions were conducted using the LLM-based

Table 1: Demographic information, including years of experience at company and with fault localization, levels of granularity that they typically work at, which faults they analyzed in the experiment, and whether they assessed that fault manually or with the LLM-based tool.

ID	Experience		Test Level	Fault ID	Treatment
	Comp.	Fault Loc.			
P1	9	13	All levels	F1	AI
				F2	Manual
P2	3.5	18	All levels	F1	Manual
				F2	AI
P3	3.5	3.5	Component / Domain	F3	AI
P4	2.5	1.5	Component / Domain	F3	Manual
P5	6.5	5.5	Domain	F4	AI
P6	3	3	Domain	F4	Manual
P7	1	3	Complete	F5	AI
P8	3	3	Complete	F5	Manual

log summarization tool, while the others were conducted without the tool, reflecting a traditional fault localization workflow.

Due to the specialized domain knowledge required for fault localization, it was not feasible to find more than two practitioners who could effectively work on the same issue. Therefore, we assigned one issue to each pair of participants based on their overlapping area of expertise, following a between-subjects design. Participants P1 and P2 were an exception, as they worked on the same fault but under different treatments. To maintain the integrity of the analysis, their data points were considered independently in the quantitative evaluation (Section 7 discusses threats to validity).

Each session took approximately 2.5 hours, or shorter if participants finished the localization task sooner. Each session was recorded and automatically transcribed using Microsoft Teams. A researcher was present during each session and took notes.

Think-Aloud Observation: During the session, each participant attempted to localize a real fault while explaining their thought process to the observing researchers as they conducted the task. Participants were given one hour to perform the localization task, and were informed that it was not important for them to finish the task—it was more important to characterize their process.

Before the think-aloud session, each participant was asked a set of demographic questions. They were also asked questions about their expectations and prior experience with LLM-based tools (see Table 3). Participants were then simply asked to carry out their workflow as they normally would when localizing a fault. In sessions where the LLM-based log summarization tool was used, the summaries for the logs related to the issue were generated before the session and provided to the participant. In these sessions, they were asked to read the summary for each log. They then were asked to provide their thoughts on the quality, accuracy and helpfulness of the summary. They then proceeded to localize the fault, with access to both the summary and raw log files.

At the end of the allotted time, they were asked to draft a fault report with as much information as they could gather during the session. This fault report was then compared with those of other participants analyzing the same fault. Afterwards, they were given 5–10 minutes to reflect, where they were asked to explain their thought process and motivations while performing the tasks.

Each fault was analyzed by a pair of participants, with one using LLM-generated summaries and the other working without them.

Table 2: Log types and sizes for analyzed faults.

ID	Log Type	Log Size
Fault 1	DLT	16.40 MB
	OSB	47.70 MB
Fault 2	DLT	46.90 MB
Fault 3	DLT	7.68 MB
Fault 4	DLT	489.00 MB
Fault 5	DLT	726.90 MB

To ensure the realism of the exercise, we did not pre-select faults. Instead, participants were instructed to book a session as soon as an actual test failure occurred, before beginning fault localization.

The analyzed faults needed to meet three criteria: they had to include log files in at least one of the supported formats, the participants could not have prior exposure to the fault, and the fault had to be representative of issues they would typically analyze. Table 2 lists the selected faults, along with their log types and sizes.

Measurement of Cognitive Load: We assess cognitive load using the NASA Task Load Index, a multi-dimensional workload measurement [17]. The framework divides productivity into six dimensions: (1) mental demand, (2) physical demand, (3) temporal demand, (4) satisfaction with own performance, (5) perceived effort, and (6), perceived frustration. Physical demand was omitted for this study as it is not applicable during fault localization. Measurement of NASA TLX for each participant requires three steps:

(1) Each participant assigns a weight to a dimension by comparing each pair of dimensions and choosing the dimension that they perceive as most contributing to their cognitive load. For instance, a participant might think that temporal demand has more impact in their cognitive load than perceived effort. Then, the weight for each dimension is the number of times it was picked over another.
(2) After completing the task, the participant rates their experience of each dimension on a scale of 0–100.
(3) Each rating is multiplied by the weight and summed. Then, the sum is divided by 10. The resulting value represents the cognitive load of the fault localization task.

Semi-Structured Interviews: The interview guide can be found in Table 3. The questions were designed to gather insights into the users' expectations and experiences regarding the fault localization process as well as the tool's functionality, usability, and impact on their workflow. When needed, follow-up questions were asked to further explore certain topics. Users were asked different questions during portions of the interview based on whether they used the LLM-generated summaries or not, enabling us to better examine each workflow in isolation.

4.2 Data Analysis

We analyze **RQ1** by using visualization and descriptive statistics of the NASA TLX values—both the final merged score and the assessment of the individual dimensions. We answer **RQ2** and **RQ3** using thematic analysis of the interview responses, following Braun and Clarke's guidelines [7]. During the coding process, we highlighted relevant parts of the transcripts and assigned code labels, i.e., short identifiers, to each. We then developed codes that describe each highlighted segment. We had multiple discussions

Table 3: Interview Guide.

	Demographic Questions (Asked Before Think-Aloud)
1	How many years have you worked at the company?
2	How many years of experience do you have localizing faults?
3	How often do you analyze logs?
4	What testing levels do you work with?
	Tool Expectations Questions (Asked Before Think-Aloud)
5	What is your attitude towards AI-assisted tools in your work?
6	What is your attitude towards LLM-based log summarization?
7	Have you used LLMs or LLM-based tools previously?
8	Do you think generated log summaries would help you in your workflow?
9	What information do you expect summaries to contain to aid you?
10	How would you use the generated summaries as part of your workflow?
	Fault Localization Questions
11	Were you able to localize the fault?
11a	If yes, what was the cause? During which activity did you find it?
11b	If not, what do you know about the fault so far?
	What do you need to be able to find it?
	Are you missing any information?
	Satisfaction Questions
12	Did you ever feel frustrated?
12a	If yes, what was the most frustrating moment?
	Questions for Sessions *Without* LLM Assistance
13	Did you ever feel stuck and unable to continue?
13a	If yes, how did you overcome this?
14	What steps in the manual process did you find most time consuming?
15	What tools or resources did you rely on the most during your analysis?
16	What does not work about the existing process of analyzing logs?
16a	How do you think these issues can be improved or overcome?
17	What was the most difficult activity you did during the analysis?
18	Was there an activity where generated summaries would have helped?
18a	Do you think the analysis could be done faster with AI assistance?
	Questions for Sessions *With* LLM Assistance
19	Do you feel like this tool made your work any easier? Why or why not?
20	How much time would it take you to analyze the logs manually?
20a	Do you feel like the summaries helped you save time? If so, how much?
21	Would this be a harder task if you did not have the summaries?
22	Would you recommend the use of LLM-generated summaries to a colleague?
23	Would you use generated summaries in your work as they are today?
23a	How would you incorporate them into your workflow?
24	Did the generated summaries help you write a better fault report?
	Questions About LLM Summary Quality
25	Were the generated summaries helpful?
26	Did the generated summaries live up to your expectations?
26a	If yes, how? What worked best?
26b	If not, why not?
27	Did the summaries provide you with all the information you needed?
27a	If not, what was missing?
28	How accurate do you think the LLM-generated log summaries were?
29	Did you ever feel misled by the generated summaries?

and iterations of the codes and code labels grouping them into themes and sub-themes.

The qualitative analysis was completed by the first two authors, with feedback from the other authors. To ensure ensure agreement in the coding, the first two authors independently coded the same interview and then assessed their level of agreement as the number of shared codes divided by the total number of codes. This resulted in a score of 0.77, which was considered sufficient to continue the thematic analysis independently.

We conducted a thematic analysis of the interview data only, as the interviews contained more detailed and nuanced reflections relevant to the research questions. In contrast, the think-aloud sessions focused mainly on the localization of specific faults. However, observations from the think-aloud sessions were used to complement the thematic analysis when appropriate.

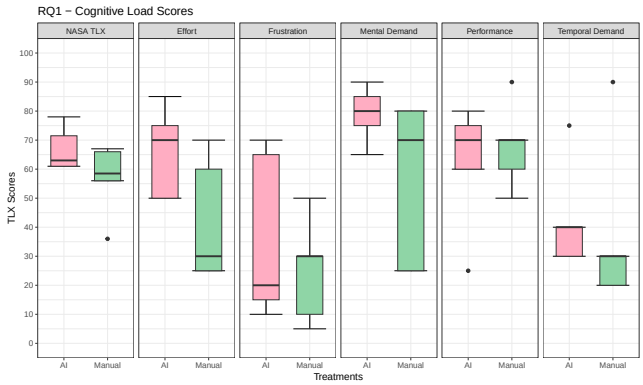

Figure 4: Box plots of the NASA TLX cognitive load, and the unweighted values for each dimension.

Data Availability: The data used for our analysis is shared in an anonymised Zenodo package [4][1]. We share the CSV files and scripts used to measure cognitive load using the NASA TLX index, as well as the interview guide and all labeled quotes from the qualitative analysis. However, due to a Non-Disclosure Agreement with our industry partner, we cannot share transcripts of the sessions or the failure logs used by our participants.

5 Results

5.1 Cognitive Load Measurement (RQ1)

NASA TLX offers a method of assessing a score for cognitive load (**RQ1**) of a task by aggregating five dimensions: mental demand, temporal demand, perceived performance, perceived effort, and frustration which are aggregated into single score. As we only had eight participants, our quantitative results cannot be assessed with statistical significance. However, the results can still inform future research. That said, we emphasize the importance of validating these results with a larger pool of participants in future work.

We plot the final TLX score for each participant in Figure 4, with higher values representing a greater cognitive load. Despite an initial hypothesis that LLM assistance could assist in log analysis, the users of the LLM-generated summaries actually reported *a somewhat higher* cognitive load (median of 63.00 versus 58.50).

To further explore the topic of cognitive load, we plot the unweighted values[2] for each individual dimension in Figure 4. In terms of median perception, the sessions using LLM-generated summaries yielded higher mental demand (80.00 versus 70.00), temporal demand (40.00 versus 30.00), and perceived effort (65.00 versus 30.00). However, the participants in the LLM and manual sessions noted the same median perceived performance (70.00), indicating that they did not feel that the summaries resulted in worse or better final results. Additionally, participants noted reduced frustration in the median

[1] https://zenodo.org/records/17493425

[2] Because each participant generates their own weight for each dimension, we use the unweighted values to more clearly compare the perceptions of participants.

Table 4: Main themes identified in the thematic analysis, with a short description of each.

Theme	Description
General Observations	Observations from the think-aloud observations.
Hindrances for Adoption	What can hinder the tool from being used.
User Observations	Participants' initial thoughts on the tool.
Improving the Tool	Aspects of the tool that could be improved.
Log Analysis Process	How the tool can improve the log analysis process.
Improving the Workflow	How the current workflow could be improved.

case (20.00 versus 30.00) for LLM-assisted sessions—although the third quartile value was worse for LLM-assisted sessions.

5.2 Thematic Analysis (RQ2–RQ3)

Thematic analysis of the interviews, supplemented with observations from the think-aloud sessions, can help assess user satisfaction (**RQ2**) and impact on workflow (**RQ3**), as well as offer further insight on cognitive load (**RQ1**). The thematic analysis produced six main themes, explained in Table 4, and a further 19 sub-themes (indicated in *italics* below).

General Observations: Most of the participants only looked at summary near the start of the session, not returning to the summaries after they begun to inspect the raw log files. Several participants also decided to first look at the log itself, before reading the summary. In some cases, participants even needed a reminder to check the summaries, as they had immediately settled into their standard manual workflow. These observations indicate the *difficulty of trying to adjust a long-standing, familiar workflow*. Even if a changed workflow may offer benefits, it may be difficult for developers to seek or adopt those changes.

Some study participants also sought additional information on how the summaries were generated, e.g., the prompts or context offered as input (beyond the raw log files). One issue raised by participants regards the option of filtering logs. In the manual workflow, the participants rarely inspect the entirety of the log. Rather, they use contextual keyword searches to filter the logs, and only inspect the sections that match the filters. The LLM-based log summarization tool also offers basic settings for filtering—and could summarize a filtered log—but in the study, we did not apply any of those settings. Therefore, the summaries often contained irrelevant details. We offered participants the opportunity to generate new summaries during the session. All participants who chose to generate new, filtered summaries found them more useful than the unfiltered summaries. However, not all participants took advantage of this option. This observation highlights the need to be able to adapt a tool to *match the user's context and workflow*.

Hindrances for Adoption: This theme encompasses issues that could hinder the adoption of this, or similar, LLM-based log analysis tools. First, participants worried that the summaries *could end up misleading users* if they do not have enough experience with the log analysis process or the tool itself. Participants gave two reasons for this conclusion. First, the tool may fail to omit irrelevant information, and second, the tool is not able to assess the importance of specific log messages. The participants observed that they possess experience and domain knowledge that the LLM does not.

> "The tool can draw conclusions that are not important. I saw that it drew conclusions that said "this is bad" or "this is not correct", I recognize that as something that is normal behavior, but if I wasn't aware of that, maybe I would be misled." - P1

Participants indicated that *the tool has a learning curve*, i.e., it may take time and usage before the user learns to integrate the tool into their workflow or to understand what they should expect of the summaries. Users may need to learn what types of faults the tool can detect from the logs, and what it is likely to miss. By developing this intuition, users would gain more confidence in the conclusions they draw from reading the summaries and understand when they need to manually inspect the logs.

> "Over time, you would know what you can expect of the tool, to trust it. It's that you really know how it works and understand the results and then it becomes more like a dialogue instead of someone just telling me this is what I know, then you can recognize this? Hopefully it will be a tool that I can use." - P1

A few participants indicated concerns that the tool may produce summaries that *may not be trustworthy*, i.e., might be missing crucial details or include misleading recommendations. This was mostly due to the irrelevant information in the unfiltered summaries.

User Observations: This theme relates to participants' initial thoughts of, and attitudes towards, the tool. Overall—despite the generally-worse cognitive load skills—participants generally offered positive and hopeful impressions.

First, the participants generally agreed that LLM-based log summarization has *low or no risk of negatively impacting* the effectiveness of the log analysis process if implemented. Even if the summary was inaccurate or incomplete, it could be read and discarded quickly, having a neutral impact. If the summary was correct and complete, then the potential positive impact is large.

> "If it is good, then I know what I should look for and be quicker at pinpointing it in the log. If it isn't, then I need to do my job anyway. So, I don't lose that much, I only see that I gain stuff." - P5

> "I guess it's a win-win. Either you still have to look manually or you get a good boost and save a lot of time and know where to start to look as well." - P3

Most participants expressed that LLM-generate log summaries have *great potential to reduce analysis time* in the long run, after overcoming the initial learning curve—especially for large logs or complex faults. While participants noted the current limitations, such as the summaries including irrelevant information and lack of domain-specific context, participants are also *hopeful that the tool will improve* over time after being integrated into practice.

> "I am 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 ... It's all a matter of how you train the model and the inputs to the model, but I have a great open mindset to that and I hope that it can be useful." - P2

Improving the Tool: Participants unanimously agreed that context and domain knowledge are needed to draw meaningful conclusions—i.e., that an out-of-the-box LLM will be limited in its capabilities for analyzing logs from complex domain-specific products that are not represented in the open-source data used to train the model.

Participants agreed that the tool need to be able to produce a *deeper analysis of the failure* to be able to give relevant suggestions. There were concern raised that the tool *lacks proprietary knowledge* of the codebase, documentation, or tests from the organization:

> "What was it missing? Preconditions, I think. Like a simple understanding of what a test is supposed to do." - P1

> "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. If you search for log analysis tools on the internet, it's almost always DevOps or some kind of server." - P1

The participants also agreed that the tool *includes irrelevant information* in the current summary. For example, logs could include messages from code elements unrelated to the elements that failed. Better filtering and prioritization mechanisms are needed.

> "It's very agnostic. It assumes that everything that it sees is more or less equal to everything else." - P1

A majority of the participants agreed that the tool *should be more interactive*. This could be done by allowing more configuration and incorporation of context before execution, or by offering an iterative process—either a chat-based interface where the user can query the LLM or through multiple rounds of execution of the tool with additional configuration in each round.

> "If I were able to set up my own prompt as a precondition? Yeah. First you have automatic summary, and if I think maybe this is the area, go into the tool, edit the prompt, run it again and see if I get a better result and give it a few attempts." - P1

> "If there's an interface where I could say 'at this time range we [action]' and then it finds and confirms 'yeah that occurred, that's here'. That would be nice." - P5

Log Analysis Process: The participants agreed that the tool may improve the overall log analysis process, reducing the time from when a participant starts to analyze logs to when a fault report is created. The summary can serve as a valuable starting point for the *drafting of the report*:

> "If I see a fatal log like this and the summary says it's a fatal log, I'll probably start to write [the report] in 10 minutes instead of later. Maybe I wouldn't submit it immediately, but I would still have a draft." - P1

Similarly, the summary offers a *first glance at the cause of the failure*, or an indication of where or what to look for, even if it does not offer a deep understanding of the issue.

Improving the Workflow: Participants highlighted activities in their existing workflow that could be improved using this tool. First, participants explained that there is too much information to process in the manual analysis, and that, despite the initial TLX results, *cognitive load could be reduced* by LLM-based log summarization:

> "I think it's partly information overload. There's so much logging. I have thousands of rows, and since we don't have the complete system, we have parts of the system simulated or missing and get a lot of errors that are expected." - P8

> "It would be good to have a short summary, at least of what the tool thinks the problem is, and also highlighting some interesting lines in the log that the tool used when it made the conclusion." - P2

Several participants stated that *on-boarding is difficult* and the hand-over for new testers is very hard. There is not much documentation and most knowledge comes from hands-on experience. They expressed hope that the tool could help reduce the initial difficulty.

> "The learning curve for the newcomers is just expanding more and more because we are working with many different things. It becomes time consuming so it's mostly hands-on practice." - P6

> "It might be unfeasible because a system like this is complex. It changes several times per year and then you have to keep everything updated. It's probably very hard. Maybe the developers have some internal documentation for what they do, but ... that's up to someone like me to understand. That someone doesn't have the same set of skills to understand the system." - P8

Almost all participants explained that they often experience *interruptions in their work* when analyzing logs. The most common is the need to ask colleagues—developers or other testers—questions.

> "It's a flawed system like in that sense. Sometimes there's no other help than them to get someone to help you" - P8

A majority raised complaints about lack of consistency or adherence to established logging standards, leading to *low log quality*.

> "All applications have their own way of writing the logs ... also, the severity levels. It's not uncommon that 'fatal' logs are 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." - P1

Several participants expressed that log analysis is *time-consuming and repetitive*, affecting their productivity and motivation. The process includes many steps that must be performed repeatedly and require very little intelligence. Participants expressed a desire to streamline or fully automate those aspects of the process, focusing their attention on tasks that require human creativity.

> "Downloading logs and manually opening them, enabling filters and searching. That's something that could ideally be automated ... So, those are rather repetitive. There are some steps you do each time. Especially if you have maybe several retries and you have to do it several times and load all the filters again." - P8

6 Discussion

Cognitive Load (RQ1): Our initial hypothesis was that LLM-generated summaries could decrease cognitive load of testers involved in the log analysis process by reducing the need to inspect the manual logs. The TLX results and observations from the the think-aloud sessions illustrate a more nuanced reality.

RQ1: Initial results suggests that the inclusion of summaries somewhat *increased* the cognitive load (8% increase in median), with the largest increases in perceived effort (117% median increase), temporal demand (33%), and mental demand (14%).

Recent studies suggest that most AI integrations fail in practice [9], suggesting that—although LLMs can help to partially automate or support tasks—the integration of LLM-based tools must be done with care. There are four observations that we make that help contextualize our results.

First, the use of generated summaries is *new to the testers*. Although we did not ask testes to interact with the tool directly, they still did not necessarily know when or how to use the summaries. In other words, education is needed on (i) how to interpret the summaries, (ii) when to use (or ignore) them, (iii) when further inspection of the raw logs is needed, and (iv) what types of events the LLM is likely to highlight or ignore.

In some cases, the testers immediately settled into their standard workflow and needed to be reminded about the existence of the summaries. *Changing an existing workflow can be difficult and cause friction*, even if there are potential benefits to doing so.

Recommendation 1: Changes to existing workflows must be motivated with clear benefits, may require time to accomplish, and trust in new tools must be earned.

Fundamentally, the tool must be accurate. Participants pointed out that the summaries contained irrelevant details. One reason for this is that *the model lacked context* on automotive software. LLMs are trained on open-source data, and it is likely that very little automotive software exists in the training sets of existing models. One way to incorporate such context is through technological means, including fine-tuning, tool access, or access to a knowledge base (e.g., using retrieval-augmented generation). This would ensure that the model has access to existing organizational context.

Recommendation 2: General-purpose LLMs will face limitations when applied in specialized domains. Ensure that models can access domain-relevant contextual information, e.g., code, tests, requirements, fault reports, and standards.

User Satisfaction (RQ2): The interviews indicated that users were not fully satisfied with the initial tool, but overall, were hopeful that it would improve in the future.

RQ2: User satisfaction was primarily affected by the level of trust in the tool, the accuracy of the summaries, and the user experience when working with the tool.

The participants generally lacked trust in the tool, due to both its current quality and mixed experiences when working with LLMs for other tasks. Irrelevant details in the summaries affected both their cognitive load and their satisfaction, and the recommendations offered previously will also affect user satisfaction. Improving the quality of the summaries is clearly a crucial step.

One aspect of user satisfaction that can be further explored is the *user experience*. Participants appreciated the initial summaries but expressed a clear interest in more interactive features. Many wanted the option to generate new summaries later in the process, once they had a better understanding of the issue, and expected these new summaries to reflect their evolving insights. Since the intention of the LLM-based log summarization tool is not to replace testers but to enhance their workflow, it is important that it can be used *flexibly and interactively*. The requests for interactivity primarily took three forms.

First, participants desired more control over the input and contextual information used to generate summaries. Second, participants would have liked to be able to ask questions about the summary to get more specific information. Third, they would like to filter the logs to facilitate the generation of more focused summaries—e.g., filtering based on an error description or specific events or program elements that they knew were relevant. The most common suggestion for how to implement these interactions was to use a chat interface, where the tester provides input using natural language.

Recommendation 3: The tool should provide a chat-based user interface where users can provide context, control or filter the input used to generate summaries, and ask follow-up questions as part of an iterative, human-in-the-loop workflow.

Workflow Impact (RQ3): During the interviews, participants noted multiple negative aspects of the current workflow, and hoped that the LLM-based log summarization tools could partially address some of these aspects.

RQ3: Participants noted that the current manual process has a high cognitive load—largely due to the quantity of information that must be processed, a difficult on-boarding process, frequent interruptions due to the need to ask colleagues for information, issues with low-quality logs, and repetitive tasks that do not require human intelligence.

The participants believed that, even if the summary quality improves, log analysis will remain primarily a human-driven process that requires creativity, experience, and intuition. However, they also expressed hope that the generated summaries could still reduce cognitive load by offering a *starting point* for the analysis process.

Recommendation 4: Generated summaries should be used early in the analysis process to highlight areas for further inspection and as a starting point for drafting a fault report.

The need to ask colleagues for information creates interruptions and causes delays. LLMs offer a flexible way to query project-specific information, as the models can analyze textual data provided as input or that is part of their training data. Therefore, if earlier recommendations are implemented (e.g., access to project and organization knowledge and a chat-based interface) it could be possible to pose questions to the LLM about aspects of the logs, source code, documentation, or test cases. Such a tool would not be a full replacement for communication with colleagues, but could reduce the number of interruptions to the log analysis process.

Recommendation 5: An LLM-based tool equipped with a chat interface and access to project and organizational knowledge could be used to reduce the number of question posed to human colleagues.

Participants also noted that log analysis involves many time-consuming and repetitive steps, such as locating, downloading, and opening files in different tools, as well as applying specific filters. They expressed a clear interest in reducing this manual effort through greater automation.

Recommendation 6: Develop an integrated log analysis environment that supports multiple log types within a single interface and automates routine steps. Such an environment could incorporate predictive features to assist common actions.

Finally, even if the LLM-based tool is improved, its effectiveness—and that of humans involved in log analysis—still depends on the quality of the underlying logs. Low quality logs will yield poor analysis results. For example, participants noted that it was common to see log messages inaccurately categorized as "error" when they should be labeled "warning" or "info".

Recommendation 7: Implement and enforce guidelines for logging to aid both humans and tools during analysis.

7 Threats to Validity

At Volvo Cars AB, each vehicle component requires specific expertise to analyze. There are few available testers with the required expertise in each area. Because we worked with actual faults instead of archived ones, we were constrained in the number of participants and faults in the experiment, which hinders **internal validity**. Moreover, the quality of LLM-based summaries and the impact of LLM-based summarization on developers may also vary across domain area and level of experience. We attempted to mitigate those limitations by sampling across multiple areas and identifying practitioners with varying experience.

The complexity of log analysis can vary between participants. However, as we asked participants to analyze newly-occurring test failures, our evaluation is representative of a randomized set of real-world faults. The participants had no previous experience with the summarization tool or its summaries. Their experience is biased by the novelty of this form of support, i.e., a **novelty effect**. We limited the impact by generating summaries in advance, i.e., practitioners did not have to learn how to interact with the tool. This enabled a clearer understanding of the impact of summaries (and not the specific tool implementation) on the log analysis workflow. However, future studies should consider the long-term impact of LLM support on the log analysis process.

Since the issues were selected at random, we had limited control over their properties, which introduces **construct validity** threats. As shown in Section 4, the largest logs in our study were 726.9 MB and 489 MB, two medium-sized issues involved logs between 15 and 50 MB, and the smallest was 7.68 MB. The evaluated size range are the most common in the industry, with larger (gigabyte-scale) logs occurring less frequently. Moreover, the variation in log sizes and levels of testing across the selected issues offers a meaningful sample for evaluating the tool's performance. The consistency observed across these cases suggests that the findings are stable within the typical log size spectrum.

Lastly, this case study is centered on participants from a single company, and the tool currently supports two specific types of logs, which introduces **external validity** threats. However, since Volvo Cars AB operates under industry-wide safety standards and one of the log types is commonly used across the automotive sector, we argue that some of our findings are likely to surface in replications of this work at other automotive companies.

8 Related Work

Many forms of automated log analyses have been proposed in the research literature [18, 22]. Of particular relevance, machine learning has shown great promise for anomaly or fault detection [2, 12, 14], root cause analysis [10], and fault localization [23].

Because LLMs have the ability to process both natural language and source code—and because they can provide interpretable results in the form of natural language—there is increasing interest in their use in log analysis and fault localization. For example, Liu et al. proposed an LLM-based technique for log-based anomaly detection [24]. In addition, Kang et al. [19] and Yao et al.[35] have proposed LLM-based approaches that localize faults to likely source code lines based on failing test cases (i.e., not using logs).

LLMs are commonly used in many domains to summarize large volumes of text [30, 36], including summarization of software artifacts, e.g., bug reports [27, 35]. LLM-based summaries have been found to be of near-human [27] or human-competitive [36] quality in some evaluations. However, concerns have been raised regarding the factual correctness of the generated text [30].

There has been limited work to date on LLM-based log summarization. Egersdoerfer et al. proposed the use of summarization as part of LLM-based anomaly detection to overcome context window limitations [15]. In their approach, log segments are analyzed, then the summary of the current and past segments is used as input for the analysis of the next segment. This method enables anomaly detection for very large logs. Their work was an early exploration and indicated the potential for LLM-based log summarization as part of a broader log analysis process.

Our approach is similar to that of Egersdoerfer et al.—we also split logs into segments and process them individually before generating a merged summary. However, our approach uses MapReduce to process in parallel, rather than sequentially summarizing using a sliding window. In addition, our study is the first to explore LLM-based log summarization in an industrial context or to explore the impact of the integration of LLM-based log analysis tools on developer cognitive load, satisfaction, or workflow. Past studies have focused specifically on quantitative accuracy assessment, while we offer a complementary perspective focused on the human aspects of LLM-based tool support. The insights from our study can inform the design of future LLM-based log analysis tools.

9 Conclusions

This paper explored the use of LLM-based log summarization to support practitioners during fault localization in an industrial automotive context. We conducted a case study in which practitioners analyzed actual faults in a CI pipeline using both manual and LLM-assisted approaches. We gathered data through think-aloud sessions, semi-structured interviews, and cognitive load assessments.

Our findings show that using LLM-generated summaries slightly increased overall cognitive load, particularly in mental demand, temporal demand, and perceived effort. However, participants reported lower frustration levels and generally viewed the tool as a useful aid rather than a replacement for their expertise. They highlighted the need for greater interactivity, especially a chat interface, and the ability to adapt summaries as their understanding of the fault evolved. These results indicate that LLM-based tools can meaningfully support practitioners in navigating complex debugging tasks, provided they are designed to complement existing workflows rather than automate them entirely.

Acknowledgments

Support was provided by Software Center project 68, "Trustworthy and Responsible AI-Centric Test Engineering (TRACE)".

References

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

[2] Anunay Amar and Peter C Rigby. 2019. 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)*. IEEE, 140–151.

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

[4] Anonymous. 2025. *Re-analysis Package: LLM Summarization for Log Analysis*. doi:10.5281/zenodo.17493425

[5] AUTOSAR. 2019. Specification of Diagnostic Log and Trace. In *AUTOSAR R19-11*.

[6] Rohini Bisht, Selomie Kindu Ejigu, Gregory Gay, and Predrag Filipovikj. 2023. 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)*. IEEE, 131–139.

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

[8] Yihan Cao, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip S Yu, and Lichao Sun. 2023. A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt. *arXiv preprint arXiv:2303.04226* (2023).

[9] Aditya Challapally, Chris Pease, Ramesh Raskar, and Pradyumna Char. [n. d.]. The GenAI Divide: State of AI in Business 2025. https://mlq.ai/media/quarterly_decks/v0.1_State_of_AI_in_Business_2025_Report.pdf.

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

[11] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (Jan. 2008), 107–113. doi:10.1145/1327452.1327492

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

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

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

[15] Chris Egersdoerfer, Di Zhang, and Dong Dai. 2023. 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*. 315–316.

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

[17] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*. Vol. 52. Elsevier, 139–183.

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

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

[20] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems* 35 (2022), 22199–22213.

[21] Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 22199–22213. https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf

[22] Łukasz Korzeniowski and Krzysztof Goczyła. 2022. Landscape of Automated Log Analysis: A Systematic Literature Review and Mapping Study. *IEEE Access* 10 (2022), 21892–21913. doi:10.1109/ACCESS.2022.3152549

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

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

[25] Abi Noda, Margaret-Anne Storey, Nicole Forsgren, and Michaela Greiler. 2023. DevEx: What Actually Drives Productivity: The developer-centric approach to measuring and improving productivity. *Queue* 21, 2 (May 2023), 35–53. doi:10.1145/3595878

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

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

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

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

[30] 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. 2023. k large language models on medical evidence summarization. *NPJ digital medicine* 6, 1 (2023), 158.

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

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

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

[34] Chunqiu Steven Xia, Yuxiang Wei, and Lingming Zhang. 2023. Automated program repair in the era of large pre-trained language models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 1482–1494.

[35] Yao Yao et al. 2024. BugBlitz-AI: An Intelligent QA Assistant. In *2024 IEEE 15th International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 57–63. doi:10.1109/ICSESS62520.2024.10719045

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

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