

UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Software Engineering Curriculum

Maksym Avramenko

LLM-based Interface for Data-Driven Waiting Time Analysis of Business Processes

Master's Thesis (30 ECTS)

Supervisor(s): Fredrik Payman Milani, PhD
Katsiaryna Lashkevich, MSc

Tartu 2024

LLM-based Interface for Data-Driven Waiting Time Analysis of Business Processes

Abstract:

Recent advancements in process mining, a field at the intersection of data science and process management, have unveiled significant potential in analyzing business processes, particularly in analyzing waiting times between different activities to identify potential bottlenecks and inefficiencies. This thesis addresses a common limitation in existing process mining tools: their fixed analytical interfaces, which restrict dynamic interaction with data and limit user-driven analysis of the process. Fixed interfaces often hinder the ability to generate customized insights and adjust analyses in response to evolving business needs. To address this, the thesis proposes the integration of a Large Language Model-based (LLM) conversational interface with process mining tool, aiming to foster an interactive engagement with the data, allowing for personalized responses and actionable insights on reducing waiting times. This initiative aligns with recent explorations of integrating LLMs with process mining to enhance user interaction and understanding. The primary contribution of this thesis is the design, implementation, and evaluation of the LLM-based interface within process mining tool, which is anticipated to allow interactive navigation with the data plus the insights on how to reduce waiting times within the process with redesign suggestions based on event logs.

Keywords:

Process Mining, Waiting Time Analysis, Process Redesigns, Large Language Models, Conversational Interface

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

LLM-põhine liides äriprotsesside ooteaja analüüsimiseks

Lühikokkuvõte:

Hiljutised edusammud andmeteaduse ja äriprotsesside juhtimise ristumiskohas paiknevas protsessikaeve valdkonnas on avanud märkimisväärse potentsiaali äriprotsesside analüüsimisel, seda eriti tegevuste vahele jäävate ooteaegade analüüsimisel, eesmärgiga tuvastada võimalikke kitsaskohti ja ebatõhususi. See lõputöö käsitleb olemasolevate protsessikaeve tööriistade ühte tüüpilist piirangut: andmeanalüüsi fikseeritud kasutusliidesed, mis piiravad andmete dünaamilist käsitlemist ja kasutajakeskset protsessianalüüsi. Lisaks takistavad fikseeritud kasutusliidesed sageli ka protsessi spetsiifikale vastavate arusaamade teket ja analüüside kohandamist vastavalt äritegevuse muutuvatele vajadustele. Selle probleemi lahendamiseks integreerib käesolev lõputöö suurel keelemudelil põhineva vestlusliidese konkreetse protsessikaeve tööriistaga, eesmärgiga soodustada interaktiivset andmetöötlust, võimaldada isikupärastatud vastuseid, ja pakkuda rakendatavaid soovitusi ooteaegade vähendamiseks. Need eesmärgid seonduvad ühtlasi ka teiste hiljutiste uuringutega, milledes rakendatakse suuri keelemudeleid protsessikaeve tööriistades, et võimaldada tulemuste käsitlemist ja tõlgendamist. Käesoleva lõputöö peamine panus on suurel keelemudelil põhineva vestlusliidese kavandamine, rakendamine ja hindamine protsessikaeve tööriistas, mis peaks seeläbi võimaldama andmete interaktiivset navigeerimist ning lisaks pakkuma protsessi sündmuslogile tuginevat teavet selle kohta kuidas protsessi ooteaegu ümberkujundamise soovitude abil vähendada.

Võtmesõnad:

Protsesside kaevandamine, ooteaja analüüs, protsesside ümberkujundamine, suured keelemudelid, vestlusliides

CERCS: P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Contents

1	Introduction	6
2	Background and Related Work	8
2.1	Process Mining for Waiting Time Analysis	8
2.2	Kronos: Analyzing Waiting Times in Process Mining	9
2.3	Large Language Models	10
2.4	LLMs in Data Analysis and Process Mining	11
2.5	Conversational Interfaces and Chatbots	12
2.6	Related Work	13
3	Method	16
3.1	Research Goals and Motivation	17
3.2	Requirements Gathering and Prioritization Methodology	17
3.3	Iterative Development	18
3.4	Evaluation Approach	19
4	System Development and Implementation	22
4.1	Requirements Gathering	22
4.2	Implementation and Architecture	25
4.2.1	System Architecture	25
4.2.2	System Development	27
4.3	Demo	38
4.3.1	Main Page Interaction	38
4.3.2	Analysis Completion and Dashboard Interface	39
4.3.3	Conversational Interface Integration	41
5	Evaluation	45
5.1	Content Evaluation	45
5.1.1	Evaluation of Improvement Opportunities Facilitation	45
5.1.2	Evaluation of Redesign Suggestions and Their Impact	47
5.1.3	General Assessment of LLM's Redesign Suggestion Capabilities	53
5.2	Performance Evaluation	53
5.3	Summary of Evaluation Results	55
6	Discussion, Limitations, and Implications	56
6.1	Discussion	56
6.2	Implications for Research	57
6.3	Implications for Industry	57
6.4	Limitations	58

7 Conclusion	59
References	62
Appendix	63
I. Licence	63

1 Introduction

In recent years, process mining has emerged as an area at the intersection of data science and process management, showing potential in analyzing business processes [24]. An event log, crucial to this field, is a digital record of the sequence of activities in a process, capturing detailed information such as activity names, timestamps, and involved resources, thus providing the data foundation for analysis [19]. This field mainly relies on examining event log data to gain insights into process performance and providing a base for optimization efforts [17]. Specifically, waiting times between different activities within a process are commonly considered a form of waste, and their analysis can highlight inefficiencies. By investigating the causes of these waiting times, opportunities for streamlining and improving process cycle times become apparent, offering potential for enhancing operational efficiency.

Waiting times in business processes are often seen during transitions between activities, such as when a case progresses from one activity to another. These periods can be caused by various factors including handoffs between resources, resource contention, or coordination issues, all contributing to potential inefficiencies within the process. While some waiting times are inevitable, others can be mitigated with strategic process design and resource management. Waiting times in business processes, such as those due to batching when activities await grouping, resource contention when activities queue for an occupied resource, prioritization when activities are delayed for others deemed more urgent, resource unavailability when resources are not working, and extraneous factors that account for delays outside the event log’s scope, are key considerations for optimizing efficiency and reducing waste [17].

A step forward in this regard is the approach introduced in the paper by Lashkevich et al [17]. This approach is implemented through a tool named Kronos, which processes event log data, formatted in CSV, to create a detailed report. This report sorts the waiting times into five clear categories: batch processing, resource contention, prioritization, resource unavailability, and extraneous factors. The resulting output, displayed through tables, charts, and graphs, provides a thorough analysis of waiting times, aiding informed decision-making and process improvement.

However, Kronos reveals a limitation: its analytical interface is fixed. The built-in charts and visualizations, while initially insightful, are confined to what the developers have pre-determined to show. Process analysts often require the flexibility to investigate specific areas of interest within the process that are not covered by these standard visualizations. Furthermore, Kronos does not actively suggest ways to redesign or improve processes, nor does it assess the potential effects of such modifications. This points to a need for an interface that is not only adaptable and user-friendly but also equipped to provide tailored insights and proactive recommendations for process optimization.

Recent works have begun to explore the integration of Large Language Models (LLMs) with process mining, aiming to address some of the inherent complexities and

diverse skill requirements within this domain [23, 1, 2]. Despite the promise shown by LLMs in augmenting conversational process mining [23], producing outputs that are both precise and contextually relevant to the complex needs of process analysis, challenges persist. Moreover, while these models have shown a solid understanding of key process mining abstractions and a notable proficiency in interpreting both declarative and procedural process models [1], the efficient conversion of traditional and object-centric process mining artifacts into textual format remains a hurdle [2]. This limitation underscores the need for ongoing improvements in LLM capabilities. Additionally, enhancing the integration of these technologies into existing process mining frameworks requires focused efforts to ensure effectiveness and efficiency.

This thesis aims to bridge this gap by developing and evaluating an LLM-based interface for Kronos. This thesis is motivated by the need to enhance Kronos with an LLM-based conversational interface, addressing the tool’s limitations. The integration’s primary objectives are: (1) to facilitate the identification of improvement opportunities by enabling inquiries about process inefficiencies; (2) to offer and recommend relevant process redesigns aimed at reducing waiting times; and (3) to predict the expected impact of these improvements on process performance.

The methodological framework of this thesis employs Design Science methodology to guide the development of the LLM-based interface. This structured approach is crucial for creating and evaluating technological artifacts, ensuring that the development is grounded in principles that foster technological innovation and practical utility [15]. Complementing this, Agile methodology is used during the product development phase. Agile’s adaptability and collaborative nature support continuous improvement and responsiveness to change [8]. This dual approach, combining Design Science for structured development and Agile for iterative refinement, ensures that the interface is both technically robust and user-centric. The process includes planning, development, and evaluation phases, allowing for steady improvements and ensuring the interface aligns with project objectives and technological advancements.

The contribution of this thesis encompasses the design, implementation, and evaluation of the LLM-based interface within Kronos. This enhancement is anticipated to substantially improve the analytical experience for process analysts by providing a more intuitive and interactive engagement with the data. Our evaluation approach will assess the performance of the interface, verifying its ability in delivering responses, its capability to furnish textual descriptions, its proficiency in rendering visual representations such as graphs, and its insightfulness in suggesting redesign strategies to mitigate waiting times.

The rest of the thesis is structured as follows. Section 2 introduces background and related work. Section 3 presents the method. Section 4 describes the implementation, Section 5 outlines the evaluation, Section 6 describes limitations of the research Section 7 concludes the thesis.

2 Background and Related Work

This section provides an overview of concepts that are relevant to the area of research in the scope of this thesis.

2.1 Process Mining for Waiting Time Analysis

Process mining has emerged as a technology at the intersection of data science and process management, aiming to extract meaningful information from event logs generated by information systems [24]. This field leverages algorithms to reconstruct, analyze, and improve real-life business processes, offering a lens into the actual execution of these processes as opposed to how they were designed to function [25].

Process mining allows analysts to explore various business questions, such as uncovering process models, assessing performance, and monitoring compliance, with a focus on identifying delays in processes [20]. There are various ways to analyze process models. One can identify inefficiencies by performing performance analyses, such as spotting bottlenecks, or evaluate adherence to organizational policies through conformance checks. The identification and mitigation of process inefficiencies, such as prolonged waiting times between activities, demand a deeper understanding of the process context and the factors influencing performance. These delays, typically occurring during transitions between tasks, can significantly hinder process flow. Waiting times are often symptomatic of deeper process issues, such as resource bottlenecks, inefficient process design, or misaligned priorities. Tackling these issues requires not only sophisticated analytical tools but also a strategic approach to process redesign and improvement. Lashkevich et al. [17] have developed a method to classify these waiting times into five main reasons: batching, resource contention, prioritization, resource unavailability, and extraneous factors, providing a detailed framework for understanding and mitigating such delays [17].

Event logs, essential for process mining, capture detailed records of process activities. They typically include a timestamp, indicating when an event occurred, a case ID to identify the process instance, and an activity name or ID, specifying the action taken. For waiting time analysis—key to identifying inefficiencies and areas for improvement—having both start and end timestamps is particularly crucial [17]. These timestamps detail when an activity begins and ends, allowing for an accurate measurement of the duration of activities and the identification of delays between tasks. These logs serve as rich datasets, forming a network of event sequences linked by temporal order and case association, thereby enabling comprehensive analysis of business processes [25]. This relational structure allows for intricate process examination, including performance evaluation and deviation detection [26].

2.2 Kronos: Analyzing Waiting Times in Process Mining

Kronos is a dedicated tool designed for the analysis of waiting times within business processes and the tool is publicly available to the users ¹. It provides an approach to decompose waiting times in activity transitions into their causes and analyze their impact on the cycle time efficiency of the process.

The preliminary step in utilizing Kronos involves the uploading of an event log, specifically in CSV format. This log is expected to comprise essential attributes for an analysis:

- **Unique Case Identifier** to distinguish between individual cases or process instances.
- **Activity Name** indicating the specific process activity.
- **Resource** representing the entity responsible for the activity's completion.
- **Start and End Timestamps** marking the activity's duration and pivotal for calculating waiting times.

Users are tasked with accurately mapping these columns to their corresponding attributes in Kronos to facilitate a precise evaluation of the process.

Upon successful input validation, Kronos performs a comprehensive analysis including several critical steps:

1. *Activity Transition Discovery* to identify pivotal transitions between activities, highlighting potential bottlenecks.
2. *Waiting Time Cause Discovery* segments waiting times into identifiable causes, such as batching, resource contention, and others, providing a detailed insight into process inefficiencies.
3. *Waiting Time Analysis* evaluates the influence of waiting times on the Cycle Time Efficiency (CTE) of the process, pointing towards possible areas for optimization.

The insights garnered from Kronos are visualized through an interface comprising tabs and transitions selector:

- The *Tabs* outline process statistics, including cases, activities, and the quantified impact of each waiting time cause (batching, prioritization, resource contention, resource unavailability and extraneous reasons); predict the potential enhancement in process efficiency (Cycle Time Efficiency) upon mitigating identified waiting times.

¹<http://kronos.cloud.ut.ee/>

- The *Transitions Selector* delves into the specifics of waiting times for specific activity transition.

These findings can be downloaded in CSV and JSON formats, facilitating further analysis or integration into reports.

Despite its significant contributions to operational efficiency, Kronos, like many process mining tools, faces inherent limitations that could impede the depth of process analysis and optimization. These limitations span both methodological aspects and implementation challenges.

Methodologically, Kronos focuses on predefined algorithms to analyze waiting times between activities, leaving potential delays at process inception or within activities due to interruptions largely unexamined. This reliance on predefined processes means that the tool can only offer insights in a preset manner, potentially limiting analysts' perspectives on uncovering inefficiencies or exploring alternative process improvements.

Moreover, Kronos's visualization capabilities, while informative, are restricted to the graphs and tables designed by its developers. This predetermined nature of visualization tools can limit analysts from diving deeper into the process data or uncovering insights that go beyond the standard visual representations. Such constraints may obscure potential inefficiencies or waiting time causes not initially anticipated, limiting the tool's flexibility and adaptability to diverse analytical needs.

On the implementation front, Kronos struggles with processing large event logs due to a cap on data handling and minimal error-handling features, necessitating a size limit to maintain functionality. This processing limitation, combined with a lack of automated suggestions for process redesign, underscores a significant challenge in the current landscape of process mining tools: the absence of solutions that offer both exploratory visual analysis and automated recommendations for process improvement. These tools often require manual intervention for redesign, relying heavily on predefined pathways that may not fully capture the unique nuances or optimization opportunities within specific processes.

In summary, while Kronos offers valuable insights into the causes and impacts of process delays, its capabilities highlight a broader issue within process mining tools — a need for greater flexibility in analysis and more dynamic solutions for process redesign. To solve these problems, we need tools that do more than just look at data. They should also offer clear suggestions for making things better, suited to the specific needs of each business process, and not just stick to a set formula or way of showing information.

2.3 Large Language Models

Large Language Models (LLMs) represent a significant breakthrough in the field of natural language processing (NLP) and artificial intelligence (AI). These models, powered by deep learning algorithms, have shown remarkable capabilities in understanding,

generating, and interpreting human-like text, fundamentally altering the landscape of computational linguistics and AI-driven applications [18].

The development of LLMs has been marked by a series of advancements in deep learning and neural network architectures. Initially, models like RNNs (Recurrent Neural Networks) and LSTMs (Long Short-Term Memory networks) laid the groundwork by demonstrating that machines could process sequences of data, such as sentences in a paragraph. However, it was the introduction of the Transformer architecture by Vaswani et al. [27] that catalyzed the rapid progress in LLMs. Transformers enabled more efficient handling of sequences by allowing models to weigh the importance of different parts of the input data without the sequential processing limitations of RNNs and LSTMs [29].

One of the most notable LLMs, GPT (Generative Pre-trained Transformer) by OpenAI, exemplifies the capabilities of these models. Starting with GPT and evolving through GPT-2, GPT-3, and the latest, GPT-4, each iteration has expanded the model's capacity for text generation, comprehension, and context understanding [5]. GPT-3 and GPT-4, in particular, with its vast number of parameters, has set new benchmarks in the model's ability to generate text that is indistinguishable from that written by humans across a wide array of topics and styles [10].

The applications of LLMs span across various domains, demonstrating their versatility and power. In content creation, LLMs assist in writing articles, generating creative fiction, and composing poetry. In the customer service domain, they power sophisticated chatbots that offer human-like interactions. LLMs also play a crucial role in translating languages, summarizing texts, generating code in software development, and even in fields as diverse as legal analysis and biomedical research, where they help in interpreting complex documents and literature [10].

Despite their remarkable capabilities, LLMs are not without challenges. Issues such as bias in model outputs, the environmental impact of training large models, and concerns around misinformation and content authenticity have sparked discussions about ethical AI development and the responsible use of LLMs [3] [14].

As the field continues to evolve, the integration of LLMs with other technologies, like process mining tools, opens new avenues for innovation. The potential to enhance process mining with the advanced text understanding and generation capabilities of LLMs could lead to more intuitive interfaces, improved analysis tools, and ultimately, more efficient and effective process management strategies.

2.4 LLMs in Data Analysis and Process Mining

The integration of Large Language Models (LLMs) into data analysis signifies a transformative shift towards more sophisticated, intuitive, and comprehensive analytical methodologies. Particularly within the domain of process mining, the potential of LLMs to revolutionize the way data is analyzed, interpreted, and utilized is immense. This section explores the burgeoning role of LLMs in data analysis, with a particular emphasis

on their application in process mining, surveying the existing literature and projects that highlight their integration into this field [22].

LLMs, such as OpenAI's GPT series, have demonstrated capabilities in generating human-like text, understanding complex queries, and providing insightful responses across various domains. These capabilities are not just limited to text generation but extend to the analysis and interpretation of vast datasets, making them invaluable tools in the arsenal of data scientists and analysts. In process mining, where the goal is to extract actionable insights from process data (event logs), LLMs offer a new paradigm for data interaction. They enable a more natural dialogue with data, allowing users to ask sophisticated questions and receive answers that are both accurate and contextually rich [22] [27].

The application of LLMs in process mining is a relatively new yet rapidly developing area. By leveraging LLMs, analysts can query process data in natural language, breaking down the barriers often imposed by traditional data querying languages. This not only democratizes access to process insights but also significantly speeds up the analysis by allowing direct interrogation of the data in an intuitive manner. For example, analysts can ask, "What are the bottlenecks in this process?" and receive a response based on the analysis of event logs, without the need for complex query construction or manual data exploration [12] [29].

Another possible use of LLMs is the development of conversational agents specifically designed for process mining tasks. These agents use LLMs to interact with users, understand their queries about process performance, and provide insights and recommendations for process improvement. Such applications of LLMs in process mining not only highlight their potential to make data analysis more accessible and efficient but also point towards future innovations in process optimization and management [5] [27].

The future of LLMs in data analysis, and particularly in process mining, is promising. As LLMs continue to evolve, becoming more sophisticated in their understanding and generation of language, their potential applications in process mining are likely to expand. This could include more advanced predictive analytics, automated process optimization recommendations, and even the generation of actionable strategies for process improvement based on historical data analysis. The ongoing research and development in this area are crucial for realizing the full potential of LLMs in transforming process mining from a primarily descriptive tool into a prescriptive and predictive instrument for business process management [5] [12].

2.5 Conversational Interfaces and Chatbots

The advent of conversational interfaces and chatbots marks a significant evolution in the way humans interact with digital systems. These technologies, rooted in the fields of natural language processing (NLP) and artificial intelligence (AI), have transformed user

interaction paradigms across various sectors by offering more intuitive and accessible means of engaging with software applications [29].

The development of conversational interfaces and chatbots has been closely tied to advancements in NLP and machine learning. Early chatbots, such as ELIZA and PARRY, laid the foundational work for simulating human-like conversations, albeit in a limited manner. With the advent of more sophisticated AI models and computational resources, contemporary chatbots have grown significantly in their ability to understand complex queries, context, and even emotions, leading to more natural and meaningful interactions [28].

Conversational interfaces have found applications across a myriad of sectors, each leveraging the technology to enhance user experience and streamline interactions. In customer service, chatbots like Zendesk’s Answer Bot provide instant support to customers, handling inquiries and solving problems around the clock. In healthcare, conversational agents such as Woebot offer mental health support through therapeutic conversations. The finance sector has also embraced this technology, with bots like Erica from Bank of America helping customers manage their finances through conversational interactions [31] [5].

One of the key advantages of conversational interfaces is their ability to improve user interaction and make data more accessible. By allowing users to interact with systems in natural language, these interfaces reduce the learning curve and make technology accessible to a broader audience [16]. They also enable more efficient data retrieval, as users can directly ask for the information they need without navigating complex interfaces or learning specific query languages. This has significant implications for fields like process mining, where analyzing complex event logs can be made more user-friendly and intuitive through conversational interfaces [22].

Despite their progress, conversational interfaces and chatbots face challenges such as understanding diverse languages and dialects, dealing with ambiguous queries, and maintaining context over extended interactions. Future developments are expected to address these issues through more advanced AI models, better contextual understanding, and personalized interactions, further broadening the applicability and effectiveness of these technologies [3].

As conversational interfaces continue to evolve, their integration into more domains, including process mining and business analytics, is anticipated. This integration promises not only to enhance user experience but also to unlock new possibilities for data interaction and business intelligence.

2.6 Related Work

Recent advancements have shown significant promise in the fields of process mining and Large Language Models (LLMs), yet there exists a distinct gap in the literature

regarding their combined use, especially in the development of LLM-based conversational interfaces for process mining tools like Kronos. While individual studies have explored the capabilities of LLMs in data analysis and the application of AI in process mining for optimization and analytics, the integration of these technologies to enhance user interaction within process mining tools remains largely unexplored.

Recent efforts have significantly advanced the integration of Large Language Models (LLMs) with process mining, demonstrating their potential in transforming the analysis and management of business processes. The technical report by Berti et al. [4] stands out by exploring the use of LLMs, specifically GPT-4 and Google’s Bard, for abstracting process mining artifacts into textual format. Their work introduces innovative prompting strategies, including direct answering, multi-prompt answering, and the generation of database queries. This approach has shown that LLMs can effectively understand and interpret key process mining abstractions, demonstrating proficiency in both declarative and procedural models and evaluating concepts of fairness in process mining. This has significant implications for advancing the object-centric process mining discipline and facilitating more efficient fairness assessments.

Similarly, Kiran Busch et al. [6] delve into the application of prompt engineering within the Business Process Management (BPM) domain. By leveraging pre-trained LLMs, their research bypasses the need for extensive fine-tuning and large training datasets, highlighting the method’s potential to bring LLM capabilities to BPM research. Their work outlines a research agenda for using prompt engineering to address BPM challenges, recognizing both the potentials and the hurdles associated with this approach.

Furthermore, the study by Michael Grohs et al. [13] illustrates the versatility of LLMs in handling text-related BPM tasks. By applying an LLM to three exemplary tasks—mining both imperative and declarative process models from textual descriptions, and assessing process tasks’ suitability for robotic process automation—they demonstrate that LLMs can perform comparably to, or even better than, task-specific solutions. This opens new avenues for BPM research and practical application, suggesting a future where LLMs serve as a general-purpose instrument for multiple process-related problems.

These contributions collectively underscore the untapped potential of LLMs in enhancing user interaction within process mining tools and BPM at large. By enabling conversational queries and interactions, these studies pave the way for a more accessible and intuitive exploration of process data, promising significant advancements in the field.

A few endeavors, such as the development of C-4PM, a conversational agent for declarative process mining by Yago Fontenla-Seco et al. [11], illustrate the attempts to create conversational agents within the process mining domain. However, this solution have not fully utilized LLMs to facilitate a conversational interface specifically tailored for uncovering waiting times, analyzing event logs in depth, and suggesting process redesigns.

This gap signifies a promising area for innovation where conversational interfaces,

powered by the intuitive and user-friendly nature of LLMs, could democratize access to process mining insights. Such advancements would not only make these insights more accessible to non-expert users but also enable a deeper and more natural exploration of process data. The potential for LLMs to significantly improve how users query and interact with process data suggests a fertile ground for future research, aiming to bridge this gap and fully realize the synergy between LLMs and process mining technologies. While existing work focuses on using LLMs for abstracting process mining artifacts, prompt engineering, and task-specific solutions in BPM, this thesis integrates an LLM-based conversational interface within a process mining tool to enhance user interaction, identify inefficiencies, and suggest process redesigns based on detailed event log analysis.

3 Method

This thesis is primarily guided by the Design Science methodology, a framework crucial for studies that propose and evaluate technological artifacts, including algorithms, models, languages, methods, systems, tools, and other computer-based technologies [15]. Design Science is chosen for its structured approach to creating and assessing innovative solutions, ensuring that the development is systematically grounded in principles that drive technological innovation and practical utility [15]. Complementing Design Science, the Agile methodology is employed in the product development phase. Agile is valued for its adaptability, collaborative nature, customer-centric approach, and facilitation of continuous improvement and responsiveness to change [8]. This dual approach, with Design Science as the foundation and Agile for iterative product development, ensures that the process is not only efficient and responsive but also aligned with rigorous academic and practical standards.

The methodology adopted in this project is visualized in Figure 1, which illustrates an approach taken from the initial analysis through to the final evaluations. This diagram encapsulates the research process, starting with the analysis of related work and defining research goals, followed by requirement elicitation and prioritization. Subsequent phases include defining the implementation strategy, undertaking iterative development and testing, and conducting comprehensive evaluations to assess accuracy, redesign suggestions, and performance.

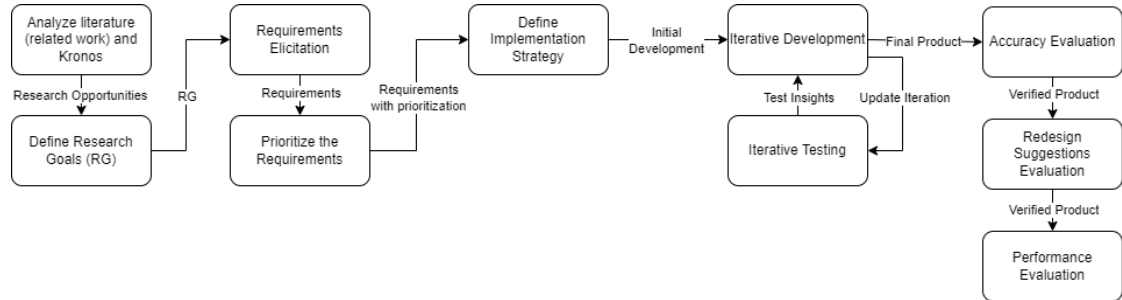


Figure 1. Methodological approach to the development and evaluation of the conversational LLM-based interface for Kronos.

In the writing of this thesis, the ChatGPT 4, developed by OpenAI ², was utilized as an auxiliary tool to assist in structuring text, rephrasing complex content, and generating LaTeX code, particularly for tables. This use of advanced language models allowed for more efficient organization and presentation of research content, ensuring that complex ideas were communicated clearly and effectively.

²<https://openai.com/chatgpt>

3.1 Research Goals and Motivation

This research aims to enhance the analytical utility of the Kronos tool by incorporating a Language Model (LLM) within its interface. The motivation stems from Kronos's current limitation of offering only a static set of visual outputs that is predefined by the developer. Research goals, which refer to a clear, concise statement that outlines the intended outcome of a study, guiding the direction and focus of the investigation [9], were defined. An LLM-based conversational interface will enable a more adaptable analysis experience for users:

1. **Facilitating the Identification of Improvement Opportunities:** The interface is designed to empower users to uncover inefficiencies by inquiring about specific aspects of event logs. The motivation here is to transition from static data representation to a dynamic interrogation process, thereby enabling users to identify and act upon improvement opportunities within their business processes.
2. **Offering Redesign Suggestions:** To address issues such as prolonged waiting times, the interface will be able to suggest feasible redesigns. The motivation for this objective is to provide users with not just data insights but also practical recommendations for process optimization.

The research is driven by the need for a more interactive tool that extends beyond static data presentation, providing a platform for users to engage with the data in a meaningful way. In essence, this thesis seeks to broaden the scope of Kronos by introducing an LLM-based interface that delivers tailored insights and recommendations, thereby enhancing the analytical capabilities of process analysts.

3.2 Requirements Gathering and Prioritization Methodology

The development of the conversational interface for Kronos is guided by a structured requirement gathering process, which is essential for aligning development with the tool's limitations. This process is informed by Kronos's existing challenges, notably its complex user interactions and static data presentation, and gaps in current research concerning conversational interfaces in process mining.

User stories are utilized to translate these needs into actionable requirements, following the format: "As a [type of user], I want [some goal], so that [some reason]." This method ensures requirements are user-centric and aligned with practical objectives. This approach is a widely recognized and commonly used method in software development [7]. User stories simplify complex needs by translating them into clear, concise narratives.

Prioritizing requirements is critical for focusing on the most impactful features first. This is achieved using the MoSCoW method by sorting functionalities into 'Must-have', 'Should-have', 'Could-have', and 'Won't-have' categories. The MoSCoW method

effectively streamlines the development focus towards essential features, ensuring the deployment of a robust initial version of the interface that addresses the most pressing user needs [30]. The prioritization categories are:

- **Must-have:** Essential for core functionality.
- **Should-have:** Important but not critical, enhance functionality.
- **Could-have:** Less critical, included if resources permit.
- **Won't-have:** Lowest priority or for future consideration.

Prioritization factors include the potential impact on user efficiency, technical feasibility, and resource availability, ensuring that development is feasible within current capabilities and project scope.

In addition, each requirement is linked to specific evaluation criteria, crucial for ensuring that development outcomes meet the intended goals. These criteria include:

- **Correctness Assessment via Event Logs:** Ensuring the interface accurately utilizes event logs for process analysis.
- **Redesign Suggestion Practicality:** Evaluating the feasibility of suggested process improvements.
- **Response Time Variability:** Assessing the interface's responsiveness across different scenarios. This linkage facilitates targeted development, focused testing, and comprehensive validation, aligning features with the interface's overall effectiveness and ensuring they address the core objectives of enhancing user interaction and data analysis capabilities in Kronos.

This streamlined method ensures the development process is not only efficient but also aligned with strategic goals, making the best use of available resources and ensuring the functionality and usability of the conversational interface.

3.3 Iterative Development

This thesis is primarily guided by the Design Science methodology, a framework crucial for studies that propose and evaluate technological artifacts, including algorithms, models, languages, methods, systems, tools, and other computer-based technologies [15]. Design Science is chosen for its structured approach to creating and assessing innovative solutions, ensuring that the development is systematically grounded in principles that drive technological innovation and practical utility [15]. Complementing Design Science, the Agile methodology is employed in the product development phase. Agile is valued

for its adaptability, collaborative nature, customer-centric approach, and facilitation of continuous improvement and responsiveness to change [8]. This dual approach, with Design Science as the foundation and Agile for iterative product development, ensures that the process is not only efficient and responsive but also aligned with rigorous academic and practical standards.

Instead of following a structured development cycle, the project will move through a series of adaptable, iterative stages. Such flexibility is crucial due to the project's dynamic nature and the ongoing emergence of new technologies. Phases within our cycles include planning, development, and evaluation, thereby adjusting the project's direction as necessary. This iterative approach facilitates a steady improvement of the interface, ensuring it stays current with technological developments and remains in line with the overall objectives of the project.

This strategy ensures that the development process is dynamic and user-centered, with a commitment to progressively refining the interface. By engaging in Agile sprints, the project remains open to adaptation and improvement, thereby fostering a robust and efficient development lifecycle.

3.4 Evaluation Approach

The evaluation of the LLM-based interface within each development cycle is essential for validating the effectiveness and efficiency of the iterative enhancements. The primary objective is to ensure that the interface meets the established criteria in terms of accuracy, speed, and utility.

Performance Evaluation Performance evaluation focuses on the operational aspects of the interface, such as responsiveness, efficiency, and error handling. The criteria for this evaluation include:

- *Response Time Variability:* We will monitor the response time of the interface, aiming for a response in under a minute. Additionally, we will assess how response time varies with different sizes of event logs, establishing benchmarks for acceptable performance across varying data volumes. This non-functional requirement ensures the tool remains efficient and responsive under varying operational loads, which is key to maintaining a positive user experience.
- *Query Processing Efficiency:* The efficiency of the interface in query processing will be evaluated. We will measure what percentage of the total response time is devoted to data retrieval as opposed to answer compilation. A reasonable maximum threshold for data retrieval time will be established to ensure balance and efficiency in processing. This non-functional requirement is vital for assessing how the

system manages its processing resources to optimize performance and user wait times.

- *Multi-Query Handling:* The interface's capability to handle multiple questions within a single prompt will be assessed. This includes measuring the accuracy and completeness of responses when faced with compound queries. This functional requirement evaluates the system's ability to understand and respond accurately to complex inquiries, highlighting its effectiveness in user interaction.
- *Error Rate in User Inputs:* The interface's error rate in interpreting user inputs will be tracked, with a focus on maintaining a low error percentage of 5%. This metric will help in evaluating the robustness of the interface's natural language processing capabilities. This functional requirement focuses on the reliability of the tool in interpreting user inputs accurately, which is crucial for trust and usability.
- *Price Per Search:* The cost-effectiveness of the tool will be evaluated by estimating the average price per search. To this end, I aim to estimate the average price per search, setting a preliminary price cap at 1 EUR per search, this consideration is essential for ensuring the tool's long-term viability and accessibility to a wide range of users. Factor influencing price will include API usage of LLMs. This non-functional requirement addresses the tool's affordability and economic feasibility, ensuring it is accessible while remaining financially sustainable.
- *Relevance to All Goals:* These performance metrics—including response time variability, query processing efficiency, multi-query handling, and error rates—are vital across all objectives. They underpin the interface's overall effectiveness, scalability, user-friendliness, and reliability, contributing to a comprehensive assessment of the system's capabilities.

Content Evaluation Content evaluation deals with the accuracy and relevance of the information provided by the interface, assessing its practicality and applicability in real-world scenarios. This includes:

- **Facilitating Improvement Opportunities:**
 - *Correctness Assessment via Event Logs:* To assess the interface's effectiveness in identifying process inefficiencies, we utilize analysis from Kronos as benchmarks. These logs serve as a factual basis to test if the interface can accurately identify areas needing improvement. Specifically, the evaluation will include involving issuing predefined questions to the LLM through a conversational interface, focusing on various aspects of waiting times within the logs. Each question will be directed to the LLM in a separate thread to

ensure unbiased responses, and correctness will be measured by comparing the LLM's answers against a manually verified baseline taken from analysis Kronos performed.

- *Relevance to Goal:* This evaluates the first goal of transitioning from static data representation to a dynamic questioning process, by verifying the data provided by the interface is correct, enabling users to spot inefficiencies.

- **Offering Redesign Suggestions:**

- *Reasonability Assessment:* The redesign suggestions offered by the interface will be evaluated for their practicality and feasibility. This assessment will determine whether the suggested process improvements are reasonable and can be effectively implemented in real-world scenarios. To ensure a thorough evaluation, interviews will be conducted with individuals possessing expertise in business process management and process mining. The selection of participants will be based on their ability to understand event logs and identify inefficiencies within processes. This expertise is essential as it enables the interviewees to assess the practicality of redesign suggestions in a real-world context and ensure that the LLM-generated recommendations are grounded in actual process dynamics and constraints. The interviews will be conducted online through Zoom. They will be recorded using Open Broadcaster Software (OBS), with immediate documentation of feedback and scores to support later analysis.
- *Relevance to Goal:* This step is critical to the second goal, ensuring the interface provides not just insights but also viable recommendations for process enhancements.

Post iteration, all findings from these evaluations will inform the decision-making process for the final implementation choice. The culmination of these iterative evaluations will lead to the selection and refinement of the most effective solution for the LLM-based interface.

4 System Development and Implementation

4.1 Requirements Gathering

Building on the methodology outlined, we developed specific requirements for the conversational interface. These requirements were derived from the identified limitations of Kronos and the gaps in current research regarding conversational interfaces in process mining. Each requirement was crafted using user stories and prioritized through the MoSCoW method to ensure they are both impactful and feasible, focusing on enhancing user interaction and dynamic data processing capabilities.

User Story 1: Interactive Query Handling

- **Story:** "As a process analyst, I want the interface to handle interactive queries about process data so that I can receive real-time insights and make informed decisions quickly."
- **Priority:** Must-have
- **Justification:** This feature is fundamental for ensuring that the interface meets the core functionality expected by users who need to interact with process data efficiently. It is essential for enhancing the user experience and the practical utility of Kronos.
- **Evaluation Link:** Directly linked to the 'Correctness Assessment via Event Logs' and 'Response Time Variability', ensuring that the interface not only responds quickly but also accurately processes queries to deliver reliable insights.

User Story 2: Intuitive User Interaction

- **Story:** "As a process analyst, I want the interface to provide intuitive interaction mechanisms so that I can use the tool without extensive training."
- **Priority:** Should-have
- **Justification:** While not critical to the core functions of Kronos, enhancing user interaction is important for broader user adoption and satisfaction. It reduces the learning curve and makes the tool more accessible.
- **Evaluation Link:** Linked to 'User Input Error Rate', which assesses how effectively the interface guides the user to make correct inputs and queries.

User Story 3: Cost-Effective Operation

- **Story:** "As a product owner, I want the interface to operate within a defined budget so that we can maintain cost efficiency."
- **Priority:** Could-have
- **Justification:** Managing operational costs is important for the sustainability of the tool, especially in a resource-constrained environment. However, this requirement is less critical than those directly affecting functionality and user experience.
- **Evaluation Link:** Linked to 'Price Per Search', ensuring that the interface remains within the financial scope of the project and organization.

User Story 4: Proactive Redesign Suggestions

- **Story:** "As a process analyst, I want the interface to proactively suggest process redesigns based on the analysis so that I can improve process efficiency."
- **Priority:** Must-have
- **Justification:** The ability to suggest actionable redesigns is critical for the utility of the conversational interface, turning insights into direct action points that can significantly impact the efficiency of processes.
- **Evaluation Link:** Directly linked to 'Reasonability Assessment' of redesign suggestions, which evaluates the practicality and feasibility of the suggestions provided by the interface.

User Story 5: Multi-Query Handling

- **Story:** "As a process analyst, I want the interface to handle multiple queries at once so that I can efficiently manage multiple issues without switching contexts."
- **Priority:** Should-have
- **Justification:** Efficient handling of multiple queries enhances productivity and user satisfaction by reducing waiting times and streamlining interactions, although it is not essential for basic functionality.
- **Evaluation Link:** Linked to 'Multi-Query Handling', which assesses how well the interface manages and responds to several inputs simultaneously.

User Story 6: Scalable Architecture

- **Story:** "As a product owner, I want the interface to be scalable so that it can grow with the increasing data volume and user base."
- **Priority:** Could-have
- **Justification:** While scalability is important for future growth and flexibility, it does not impact the immediate functionality of the interface but prepares the system for long-term success.
- **Evaluation Link:** Linked to general performance metrics like 'Response Time Variability', which will help ensure that the system maintains performance as scale increases.

User Story 7: Facilitation of Improvement Opportunities

- **Story:** "As a process analyst, I want the interface to analyze event logs and provide detailed insights into process inefficiencies, highlighting potential improvement opportunities without needing manual intervention."
- **Priority:** Must-have
- **Justification:** The core functionality of any process mining tool is to provide actionable insights that can lead to significant operational improvements. This feature is essential for ensuring that Kronos not only identifies issues but also suggests data-driven solutions automatically, enhancing the tool's practical value for continuous process optimization.
- **Evaluation Link:** This requirement is directly linked to 'Correctness Assessment via Event Logs', which will evaluate the accuracy of the insights provided by the interface, and 'Reasonability Assessment', which will measure the practicality and feasibility of the suggested improvements.

These user stories directly address the primary concern of providing detailed, actionable data about process analyses and improvements. Prioritizing these features ensures that the conversational interface for Kronos will not only enhance user interaction but also empower users with powerful tools to manage and optimize their processes efficiently. The prioritization reflects a strategic focus on functionalities that maximize immediate benefits while laying a foundation for continued innovation and user satisfaction.

4.2 Implementation and Architecture

4.2.1 System Architecture

This subsection outlines the initial architecture of Kronos and changes that have been implemented to bring conversational capabilities to Kronos, further aligning the tool with the interactive needs of its users.

The flow depicted in Figure 2 illustrates view of how user queries are processed and used in this section to showcase how the application and its interface are working. Initially, the user uploads an event log, which is then analyzed by Kronos to discover activity transitions, waiting time causes, and perform waiting time analysis. This analysis data is used to visualize the analysis results. When a user poses a question, the system, based on the LLM's request, retrieves relevant data from this analysis to inform an LLM. The LLM then processes this data to generate a response, which is subsequently output to the user as the LLM's response. The vital part of Kronos, waiting time analysis and existing functionalities, will not be changed during this work, but additions will be made so the conversational interface could be integrated seamlessly.



Figure 2. Process flow illustrating the steps from event log upload to LLM response generation.

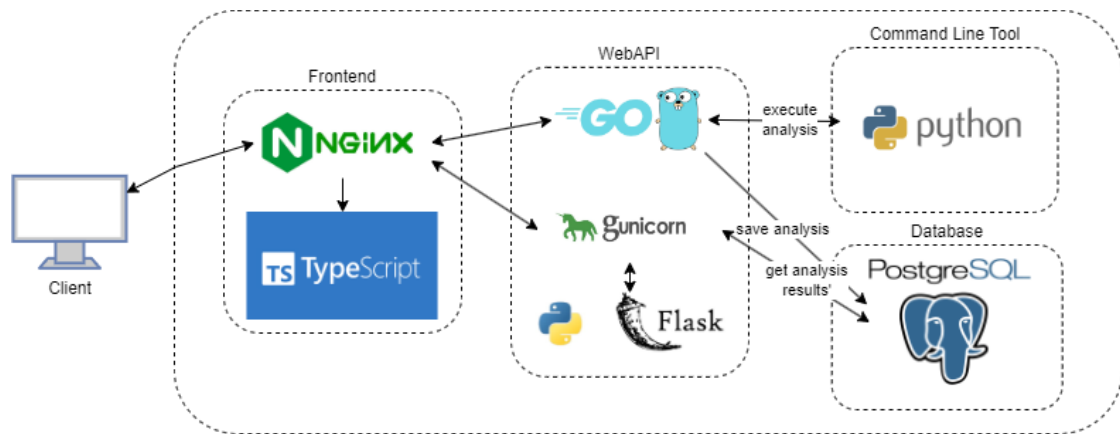


Figure 3. Architecture of Kronos Web Application

Figure 3 showcases the architecture of the Kronos Web Application from the developer point of view. The system is divided into four main dockerized services: the Frontend, Backend, Command Line Tool, and Database Service.

The Frontend, implemented using TypeScript and the Angular framework, serves as the user interface. It is where users upload event logs and engage with the system to view the results of the analysis. This service is containerized using Docker to ensure consistency across different environments and is served through an NGINX server for efficient request handling and static file serving. The source code for this component is available on GitHub³.

At the core of the application lies the Backend, which is responsible for managing job queues and processing jobs. Developed with Go, it handles incoming job requests and leverages Gunicorn, a Python WSGI HTTP Server, to manage interactions with the Command Line Tool. The Backend's dockerization guarantees isolation and swift deployment. The Backend's repository is accessible on GitHub⁴.

The Command Line Tool, written in Python, conducts the primary analysis of event logs. As a separate Docker image, it ensures modularity and scalability. Upon completion of the analysis, results are stored in a structured CSV format, ready for retrieval by the Backend. The Command Line Tool's code can be found on GitHub⁵.

PostgreSQL is used for the Database Service, where the results from the Command Line Tool are stored. Dockerization of this service facilitates straightforward version management and setup. Through this service, the Backend can access and manage the analysis data, underpinning the application's data persistence. The database set up is hosted in docker compose file as a part of Backend GitHub repository⁴.

Each of these services is a separate component within the application's dockerized ecosystem, ensuring seamless integration and high maintainability. The dockerized approach also simplifies the process of updating individual components without affecting the entire system, crucial for continuous integration and delivery practices.

The process of data analysis in the Kronos Web Application is a multi-stage operation that begins when a user interacts with the frontend and culminates in the presentation of the analyzed data. The steps are as follows:

1. **Job Submission:** When a user uploads an event log via the frontend interface, the system encapsulates this log into a new job and submits it to the backend. This action triggers the job processing sequence within the application's workflow.
2. **Job Processing:** Upon receiving a new job, the backend places it into a managed job queue. This queue helps in organizing the order and priority of jobs to be processed. The backend then dispatches the job to the Command Line Tool operating within a Docker container, dedicated to executing the event log analysis.
3. **Result Handling:** The Command Line Tool, after processing the event log, compiles the analysis results into a CSV file. This file is then uploaded to the Database

³<https://github.com/AutomatedProcessImprovement/waiting-time-frontend>

⁴<https://github.com/AutomatedProcessImprovement/waiting-time-backend>

⁵<https://github.com/AutomatedProcessImprovement/waiting-time-analysis>

Service, where it is stored for subsequent retrieval. Upon successful storage of the results, the backend updates the job status to reflect its completion.

4. **Data Retrieval:** The frontend, in an effort to provide timely feedback to the user, regularly checks the status of the job. Once it detects that a job has been marked as completed, the frontend sends a request to the backend to aggregate the data. The backend, in turn, fetches the processed data from the Database Service and sends it back to the frontend for display to the user.

Integration of the Conversational Interface To achieve the thesis goal of enhancing user interaction with Kronos through a conversational interface, strategic modifications were made to both the frontend and backend components.

A key addition to the frontend is the integration of a new component dedicated to conversational interactions. This component is seamlessly embedded within the main user interface, ensuring that users can access the conversational features without navigating away from the primary application context. The design decision to embed this feature directly in the main page facilitates a fluid user experience, as users can engage in dialogue with the system while simultaneously interacting with the analytical functionalities of Kronos.

The backend has been augmented with a new component responsible for the internal processing of chat messages. This ensures that user inputs are efficiently parsed and handled, laying the groundwork for a responsive conversational interface.

- **API Extensions:** To support the new conversational capabilities, additional endpoints have been added to the backend API. These endpoints specifically cater to the operations of the conversational interface, managing the flow of data required for processing and responding to user queries.
- **External API Interaction:** The backend has been equipped with the capability to communicate with the external OpenAI API. This allows the system to leverage the sophisticated processing of Large Language Models to generate intelligent and contextually relevant responses for the users. This external interaction is designed to be triggered as needed, based on the complexity and nature of the user's request.

4.2.2 System Development

This subsection details the developmental considerations and decisions undertaken during the creation of the conversational interface.

Selection of an LLM The initial phase of the development involved deciding between self-hosting an LLM and utilizing a commercial service. Due to the cost implications

of maintaining a dedicated server for an LLM and the relatively low volume of queries anticipated, it was deemed more cost-effective and efficient to opt for a commercial LLM. Commercial services offer the advantage of robust, high-speed processing capabilities that are scalable and financially more viable for handling sporadic query loads.

Among the available commercial LLMs, OpenAI's ChatGPT-4 Turbo was selected for its enhanced capabilities and efficiency. Released around the time of this study, GPT-4 Turbo was noted for its increased capacity and reduced operational costs. It supports an extensive 128K token context window, which is beneficial for processing complex queries that require a deeper understanding and retention of context. OpenAI highlights that GPT-4 Turbo is particularly adept at following precise instructions and generating outputs in specific formats, an essential feature for our application needs⁶.

Furthermore, OpenAI provides well-documented APIs and Python libraries that facilitate seamless integration and interaction with their models. This accessibility greatly simplified the process of incorporating the LLM into our system, ensuring that we could leverage the advanced capabilities of GPT-4 Turbo efficiently.

The decision to utilize a commercial LLM, specifically OpenAI's GPT-4 Turbo, was driven by both technical and economic considerations. The capabilities of GPT-4 Turbo in handling complex, instruction-based tasks, combined with the cost-effectiveness of using a scalable commercial platform, laid a solid foundation for the development of the conversational interface.

OpenAI's Assistants API The subsequent phase of development involved a thorough assessment of OpenAI's Assistants API to determine its suitability for our conversational interface needs. This API is specifically tailored for applications that require a dynamic and interactive assistant capable of handling complex tasks beyond simple text generation.⁶

The Assistants API by OpenAI introduces several advanced features designed to enhance the functionality and flexibility of AI-driven applications⁶. Key capabilities of this API include:

- **Code Interpreter:** This tool allows the AI to write and execute Python code within a secure, sandboxed environment. It is particularly useful for applications that require data manipulation, graph generation, or complex computational tasks.
- **Retrieval:** This feature enhances the assistant's capabilities by integrating external knowledge sources. It can access proprietary data or public information, thereby broadening the assistant's knowledge base without the need for extensive pre-processing or embedding computations.

⁶<https://openai.com/blog/new-models-and-developer-products-announced-at-devday>

- **Function Calling:** Assistants can call custom-defined functions, allowing for the integration of specific business logic or data processing routines directly within the AI's workflow.

The Assistants API supports persistent threads, allowing continuous interaction and overcoming context window limitations. This feature is crucial for applications that need sustained dialogue for complex analyses or ongoing user engagement.

The Assistants API was chosen for its versatility and ability to meet our project's needs. Its capabilities enhance our conversational interface's ability to understand complex queries, interact naturally, and perform various tasks effectively.

Preliminary Architectural Adaptations for LLM Integration The integration of OpenAI's Assistants API into the Kronos system necessitated targeted architectural changes, particularly affecting the frontend and backend components. These modifications were essential to accommodate the new conversational interface capabilities while maintaining the integrity and functionality of the existing system.

The database, responsible for storing analysis data, required no schema changes. However, it was essential to ensure that the new components had appropriate access to perform data retrieval operations. This was facilitated by the inherent connectivity provided within a Docker environment, where services are configured to communicate seamlessly.

The Command Line Tool, which executes the core analysis of event logs, remained unchanged. This decision was made to preserve the integrity of the data analysis processes that Kronos was already handling effectively.

In the frontend, the addition of a conversational interface was conceptualized to complement, not replace, the existing user interface elements. Positioned at the bottom of the main page, this new component allows users to interact with the LLM while simultaneously accessing detailed process analysis. This dual-accessibility design ensures that users can engage in dynamic conversations with the LLM without losing visibility of the ongoing analyses. The frontend development continued to utilize TypeScript, leveraging its robust features for building scalable and maintainable web applications.

Significant extensions were made to the backend to support the new functionalities. New API endpoints were developed to handle the communication between the frontend and the LLM. These endpoints facilitate the transmission of user queries to the LLM and the reception of responses. The backend's expansion included integrating with OpenAI's Assistants API to fetch and relay the LLM's outputs effectively. Python, already a part of the backend's technology stack, was further utilized due to its excellent support for OpenAI libraries, simplifying the integration process and ensuring efficient data handling.

Communication Logic between Frontend and Backend The implementation of conversational threads to maintain the history of interactions between the user and the

LLM presented a complex challenge. The adopted approach hinged on the creation of a seamless communication channel between the frontend and backend, exploiting the persistent thread feature offered by the Assistants API.

The process commences with the user inputting their initial message into the system. This action triggers an API request to the `/start` endpoint, transmitting two key pieces of information:

- **jobid**: A unique identifier for the Kronos job, facilitating subsequent data retrieval from the analysis database.
- **message**: The message composed by the user, initiating the dialogue with the LLM.

Upon receiving this information, the backend engages in a multi-step procedure:

1. It creates a new thread through the Assistants API, leveraging OpenAI's Python client library.
2. A message is then crafted within this newly formed thread.
3. Finally, a `'run'` is instantiated to process the message.

The backend subsequently relays the *thread_id* and *run_id* back to the frontend. These identifiers are critical for the frontend to request updates on the message processing status.

The frontend initiates a polling mechanism, periodically sending requests to the backend every five seconds. The backend's response can either contain the awaited LLM message or indicate the continuation of processing. This exchange leverages a Gunicorn-Flask setup, with distinct endpoints delineated for the initial message and the polling actions:

POST `/start`

GET `/status/<jobid>/<threadid>/<runid>`

On the frontend side, communication with the backend is executed using JavaScript's Fetch API with the POST method for the initial message, and the appropriate headers are set to ensure the correct content type. During this interaction, all relevant data is temporarily held in the local memory, ensuring privacy and security as the data is lost upon page refresh, which echoes the ephemeral nature of a conversation.

The diagram provided in Figure 4 encapsulates the aforementioned communication logic, offering a visual representation of the flow from the user's initial message to the backend's processing and OpenAI's Assistants API interaction.

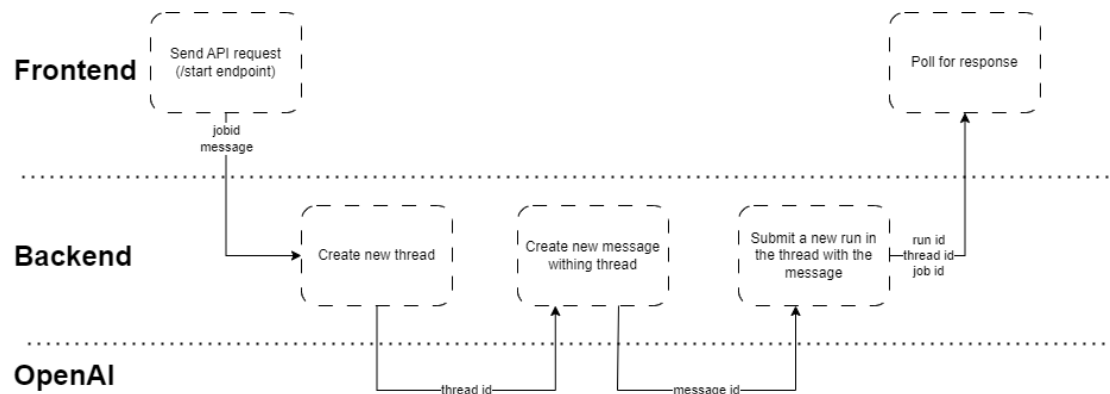


Figure 4. Flow of the initial message through the conversational interface system.

Subsequent Message Processing Building upon the foundation established for initial communications, the system is designed to maintain continuity within a conversation thread. This is crucial for the LLM to access the history of interactions and provide contextually relevant responses.

When the user submits additional messages after the first, the frontend leverages an alternative endpoint to ensure the conversation's flow within the established thread:

POST /process

The data transmitted via this request includes:

- **thread_id**: The identifier for the ongoing conversation thread.
- **jobid**: The unique identifier for the job, consistent throughout the dialogue.
- **message**: The new user message to be processed by the LLM.

Upon receiving the message continuation, the backend performs the following steps:

1. Submits the new message to the existing thread.
2. Commences a new run within the thread to process the message.

Following these actions, the backend enters a polling state, awaiting the LLM's response.

The same as to the initial message, the frontend executes a systematic polling operation at five-second intervals, querying the backend for updates on the message status. This iterative process continues until a response is received or an error is encountered.

The diagram provided in Figure 5 delineates this process, illustrating how subsequent messages are incorporated into the existing thread and processed by the system.

The implemented method ensures that if users wish to initiate a new conversation, they can simply refresh the page. This action clears the thread information from the local memory, resetting the dialogue and reverting to the use of the '/start' endpoint for a fresh interaction.

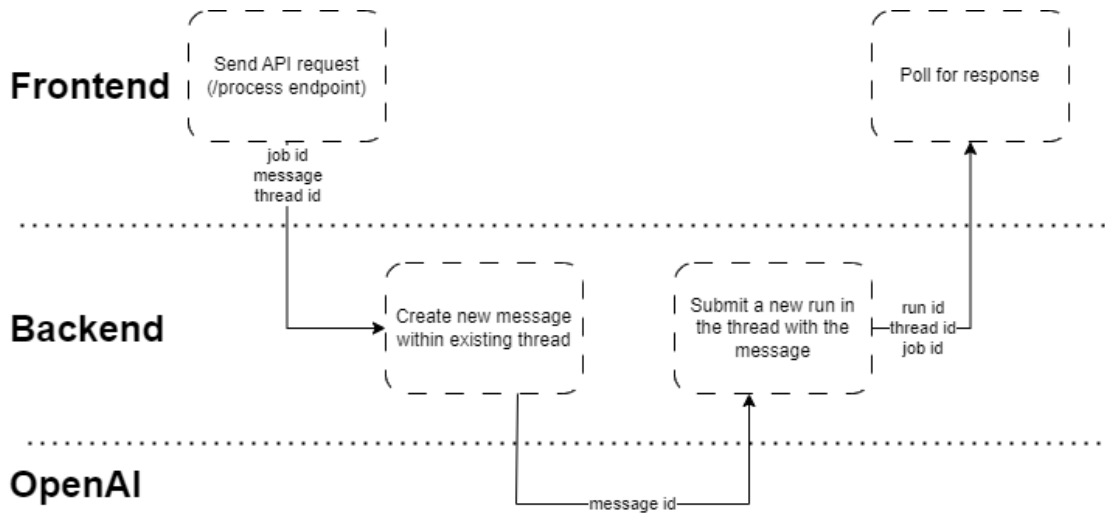


Figure 5. Procedure for handling subsequent messages within an existing conversational thread.

Polling Mechanism To facilitate the polling process, the backend provides a dedicated endpoint:

```
GET /status/<jobid>/<threadid>/<runid>
```

This endpoint expects the **jobid**, **threadid**, and **runid** as parameters to accurately fetch the status of a message being processed by the LLM.

Upon receiving a polling request, the backend employs the function provided by OpenAI's Python library. This function `client.beta.threads.runs.retrieve` is instrumental in obtaining the latest status update of a specific message within the conversational thread.

The backend discerns the message status based on the `run_status` attribute:

- If `run_status` is 'completed', the backend extracts the content of the final message from the thread's message list and returns it to the frontend along with a 'completed' status.
- If `run_status` is 'in_progress', the backend notifies the frontend to continue polling by returning an 'in_progress' status.

The chosen polling interval of five seconds provides a balance between user expectations for prompt responses and the system's capacity, preventing undue stress on the infrastructure while awaiting LLM processing completion.

Upon receipt of the 'completed' status, the frontend stops further polling requests and presents the response to the user. Conversely, an 'in_progress' status prompts the frontend to reinitiate the polling request after the designated interval.

Frontend Implementation The frontend of the conversational interface is designed to facilitate an intuitive user interaction, enabling seamless communication with the backend and, ultimately, with the LLM. The implementation details provided below outline how the frontend manages user input and the display of responses.

The chat interface on the frontend is composed of a simple structure that includes a text field for input and a 'Send' button for submitting messages. These elements are constructed using a card layout that dynamically adjusts to accommodate the conversation history.

Interaction with the chat interface is designed with user convenience in mind:

- The 'Send' button is temporarily disabled during message processing to prevent multiple submissions.
- The input field allows users to type their message and submit it either by pressing the 'Enter' key or clicking the 'Send' button.
- The size of the chat history area dynamically adjusts based on the length of the conversation to provide a streamlined view of the interaction.

To ensure that responses from the LLM retain their original formatting, a markdown rendering function is implemented on the frontend. This function translates markdown syntax into corresponding HTML, enabling the proper display of headings, bold and italic text, links, and line breaks. The core logic of the rendering function is as follows:

- Headings are converted into HTML heading tags, maintaining their hierarchical importance as specified by the markdown syntax.
- Text emphasis such as bold and italic is rendered using strong and em tags, respectively.
- Hyperlinks are preserved by transforming markdown link syntax into clickable HTML anchor tags.
- Line breaks in the LLM's response are replaced with HTML break tags to maintain the intended text layout.

To achieve that, text.replace function is used to replace all markdown heading with HTML headings.

Data Accessibility for LLM After establishing the communication between the user interface and the LLM, the next pivotal step involved granting the LLM access to process data.

Initial attempts to feed data into the LLM involved direct input of CSV files containing Kronos analysis. However, this method proved inefficient due to:

- The potentially large size of event logs, leading to extensive processing times.
- The propensity of the LLM to commit errors in Python code, resulting in inaccurate data interpretations.

The function calling feature offered by OpenAI's Assistants API was identified as a more efficient alternative. This approach involves:

- Describing the function within the API call, enabling the LLM to intelligently decide when to generate a JSON object containing function call arguments.
- Executing function calls in the user's environment, using the JSON generated by the LLM.
- The model's capability to perform parallel function calls, improving efficiency, especially with time-intensive functions.

The function 'query_database' was defined with specific characteristics to interface with the database holding Kronos analysis. This function allows the LLM to generate SQL queries, which are then processed on the backend, with the results returned to the LLM for continued analysis.

The function configuration provided to the LLM outlines the structure and scope of the SQL query function call:

```
{
  "name": "query_database",
  "description": "Executes a read-only SQL query on a specific database that contains analysis on waiting times and returns the results. The query should be limited to select statements. Use '{table_name}' as a placeholder for the actual table name in your query. Available columns: starttime, endtime, wtcontention, wtbatching, wtprioritization, wtunavailability, wtextraneous, sourceactivity, destinationactivity, sourceresource, destinationresource, wttotal.",
  "parameters": {
    "type": "object",
    "properties": {
      "query": {
        "type": "string",
        "description": "The SQL query to execute. Include '{table_name}' as a placeholder for the table name."
      }
    }
  },
  "required": ["query"]
}
```

```
}
}
```

This configuration serves as a blueprint for the LLM to understand the accessible data structure, allowing for accurate and targeted data retrieval queries.

- **Name:** 'query_database' – Identifies the function for SQL queries in the Kronos database.
- **Description:** Executes read-only 'select' SQL queries, ensuring only data retrieval without modification.
- **Placeholders:** Uses '{table_name}' as a dynamic placeholder in queries for actual table names.
- **Available Columns:** Lists columns like 'starttime' and 'endtime', aiding in forming relevant queries.
- **Parameters:**
 - Defined as an object specifying the SQL query format.
 - Includes a 'query' property, accepting a string with the SQL command.
 - The 'query' requires a placeholder for table names, enabling backend scripts to adapt it based on the job context.
- **Requirements:** Specifies that a 'query' is necessary for function execution.

In order to enrich the conversational interface's capabilities for more sophisticated insights, a suite of function descriptions was integrated additionally within the assistant model. These functions allow the LLM to access and analyze data beyond the standard analysis provided by Kronos. They include strategies for batching, prioritization, and additional resource information which are not typically present in the standard Kronos report.

Function: Discover Batching Strategies

```
{
  "name": "discover_batching_strategies",
  "description": "Returns batching strategies of an event log.
  The function internally processes a pre-loaded event log and
  discovers batching strategies, providing insights into batch
  processing instances within a process. It characterizes each
  batch with details about the activity, involved resources, batch
```

```

    processing type (sequential, concurrent, or parallel), frequency,
    batch size distribution, duration distribution, and firing rules
    for batch initiation.",
}

```

The `discover_batching_strategies` function serves to illuminate the batch processing instances within an event log, offering information into how these batches affect overall process efficiency.

Function: Discover Prioritization Strategies

```

{
  "name": "discover_prioritization_strategies",
  "description": "Discovers and returns prioritization strategies from
    an event log. This function analyzes the pre-loaded event log to
    identify case priority levels and classify process cases into
    corresponding levels. It provides insights into how cases are
    prioritized, ensuring that high-priority activities are executed
    before.",
}

```

The `discover_prioritization_strategies` function enables the LLM to deduce how prioritization within a process affects case handling and resource allocation.

Function: Discover Resource Calendars

```

{
  "name": "discover_resource_calendars",
  "description": "Discovers and returns list of resources and their
    working calendars",
}

```

The inclusion of `discover_resource_calendars` is to give the LLM insight into the working time of various resources, crucial for optimizing process flow.

Function: Discover Case Attributes

```

{
  "name": "discover_case_attributes",
  "description": "Discovers and returns case attributes present in the
    event log. If returned data is empty, then there is no case attributes
    present in the event log.",
}

```

The `discover_case_attributes` function aims to provide the LLM with additional context regarding the static characteristics of cases within the event log.

The integration of these function descriptions allows the LLM to generate more targeted and nuanced queries and responses. They do not require direct input parameters but leverage the rich data provided by Kronos' backend infrastructure, thereby enhancing the LLM's decision-making support capabilities.

When the status endpoint receives a `'requires_action'` status, the backend must process the required function(s) as per the LLM's request. These functions, identified by a list of `tool_calls`, encompass pre-defined tasks that are essential for the LLM to generate an informed response.

The frontend's role remains consistent: to regularly send a request to check for updates from the LLM. If a `'requires_action'` status has been handled, the frontend will be updated with the name of the function executed, keeping the user informed of the data sources utilized by the LLM.

Assistant's instructions To ensure that the assistant not only understands its operational context but also adheres closely to the requirements of processing event logs for waiting time analysis, facilitating targeted and effective process improvements, instructions were added to the assistant. This paragraph outlines the description of instructions provided to the assistant, but full instructions are available on the GitHub of Kronos Backend service ⁷.

Instructions provided to the assistant are tiered at multiple levels to enhance its responsiveness:

- **Assistant Instructions:** Broad guidelines about the assistant's capabilities and operational context.
- **General Message Instructions:** Specific directives accompanying each message, focusing on immediate response needs.
- **Direct Message Instructions:** Highly detailed instructions embedded within messages to refine the assistant's responses. This option is not provided by OpenAI directly, but rather a possibility for us to integrate instructions by concatenating user message and these instructions.

The assistant is designed as an expert in process mining and process improvement, integrating closely with the Kronos tool to enhance its capabilities. These instructions form the backbone of how the assistant interprets and responds to user queries.

The assistant functions as an expert within the domain of process mining, specifically tailored to work with the Kronos process-mining tool. Kronos analyses event logs to

⁷<https://github.com/AutomatedProcessImprovement/waiting-time-backend/blob/main/assistant-instructions.txt>

determine the causes of waiting times in business processes. The assistant's role is to interpret these analyses and assist in identifying and addressing inefficiencies.

The primary goal of the assistant is to facilitate the analysis of waiting time causes, unearth inefficiencies and bottlenecks, and suggest actionable process redesigns to mitigate these inefficiencies, ultimately aiming to reduce cycle times and enhance process efficiency.

- **Waiting Time:** Defined as idle time between activities, identified during transitions between sequentially executed activities.
- **Waiting Time Cause:** Direct sources of delay within a transition, decomposed into intervals attributable to specific causes like batching, prioritization, and resource contention.

The assistant leverages the Kronos report, which compiles detailed transition executions and their associated waiting times. Each transition is analyzed for potential improvement by examining the aggregated data across multiple executions.

- **Batching Strategies:** Outlines various batching approaches within the process, providing insights into batch size, frequency, and activation rules.
- **Resource Calendars and Prioritization:** Examines resource availability and prioritization strategies to suggest optimal resource allocation and scheduling adjustments.

Throughout the development phase, the focus was on ensuring that the LLM could access sufficient and relevant data, enabling it to provide insightful and actionable feedback on process inefficiencies and potential improvements. The implementation of the Assistants API from OpenAI played a crucial role in this, allowing the LLM to interact dynamically with the system and perform complex data retrieval and processing tasks effectively.

4.3 Demo

The Kronos Web Application is accessible to users ⁸, offering an interface for waiting time analysis and conversational interface.

4.3.1 Main Page Interaction

Upon visiting Kronos, the user is greeted with the main page, depicted in Figure 6. Here, users can either search for results by a previously obtained job ID or upload their own event log for analysis.

⁸<http://kronos.cloud.ut.ee/>

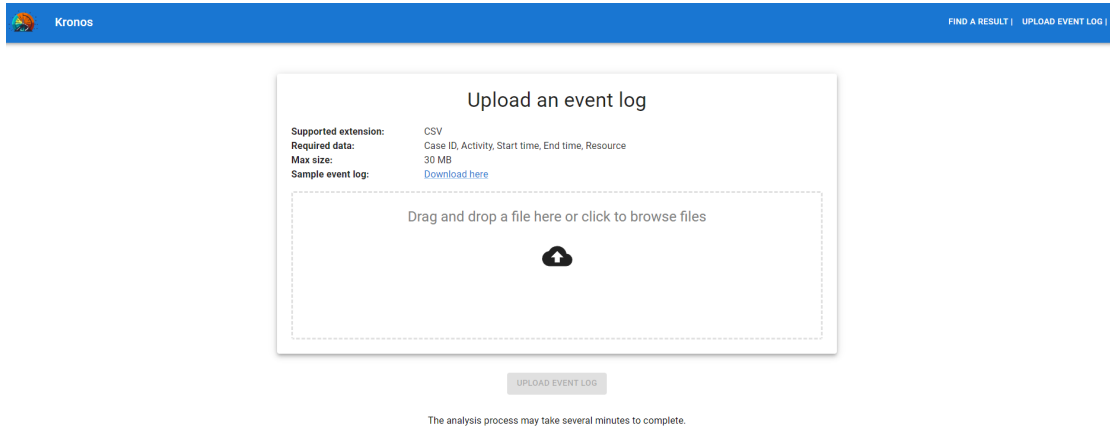


Figure 6. The main page of Kronos where users can upload event logs.

Event logs must adhere to specific criteria: they should include a Case ID, Activity, Start time, End time, and Resource, which collectively facilitate the mining and analysis of process workflows. The log file's size is limited to 30 MB and must be in CSV format. Users can download a sample event log provided on the main page for reference.

To upload an event log, users interact with the interface shown in Figure 7. Upon selecting or dragging a file to the interface, users are prompted to map the columns of their CSV file to the required fields. This ensures that each piece of essential data is correctly identified for the analysis process.

Each required field—Case ID, Activity, Start Time, End Time, and Resource—can only be mapped once to prevent data conflicts. If the event log contains additional, non-required columns, the user can assign them as 'Other'.

Once the mapping is complete, the user can initiate the analysis by clicking 'CONTINUE'. This action submits a new job to the backend, where it enters the processing queue.

Following the job submission, the frontend displays a unique job ID to the user and begins periodically checking the job's status every five seconds. When the job is completed, the frontend redirects the user to a new page where the analysis results are presented.

4.3.2 Analysis Completion and Dashboard Interface

Upon the successful processing of an event log, users are navigated to the dashboard page, which offers a comprehensive view of the analysis results. Figure 8 illustrates this interface.

Map the event log

Here you can assign the right column names to each column for processing.

case_id	<div>Type</div>
start_time	<div>None</div>
end_time	<div>Case ID</div>
Role	<div>Activity</div>
Activity	<div>Start Timestamp</div>
Resource	<div>End Timestamp</div>

[CANCEL](#) [CONTINUE](#)

Figure 7. The event log mapping interface on Kronos.

The dashboard page first presents an overview with key metrics such as the number of cases, activities, and transitions observed within the event log. It features interactive pie charts displaying the cycle time and waiting times distribution. These charts include selectable options to view data aggregations either as averages or totals across different dimensions:

- **Resource Unavailability:** Time lost when a resource is not available.
- **Resource Contention:** Time when resources are occupied with other tasks.
- **Batching:** Time pending for batch processes to complete.
- **Prioritization:** Time extended due to prioritizing other activities.
- **Extraneous Factors:** Time affected by factors outside of typical process control.

Users can delve into each specific reason for waiting times by navigating through additional tabs dedicated to each category.

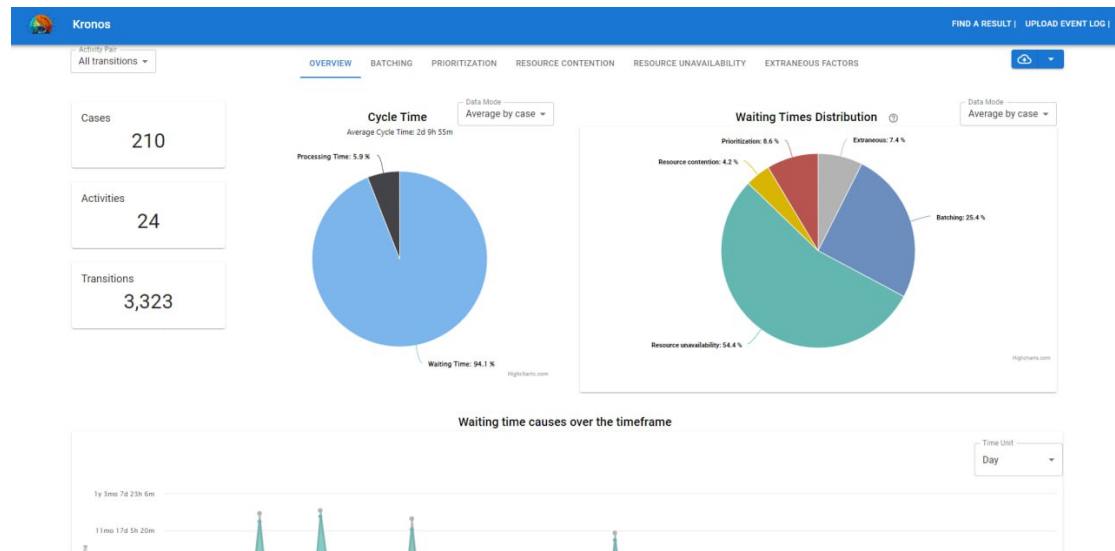


Figure 8. Dashboard page showcasing the analysis results.

A selector for activity pairs allows users to filter the data to view data relevant to specific transitions between activities. This pre-existing functionality was not modified as part of the thesis work.

4.3.3 Conversational Interface Integration

Located at the bottom of the dashboard page, the chat interface adds a conversational element to Kronos, allowing users to interact with the LLM directly. Exactly this part was added as a part of research work. It includes an input box where users can type their queries or messages and a 'SEND' button to submit their text to the LLM for processing.

Figure 9. Chat interface for LLM interaction on the dashboard page.

Upon sending a query through the chat interface, the frontend communicates with the backend by calling the `start_request` REST API endpoint. This endpoint receives the user's `jobid`, which is the identifier for the analysis job in Kronos, and the message, which is the user's query.

The backend then interacts with the OpenAI API to initiate a conversation thread, sending the user's message and creating a new run. The frontend, upon initiating this

request, displays a loading symbol, indicating that the query is being processed by the LLM, as seen in Figure 10.

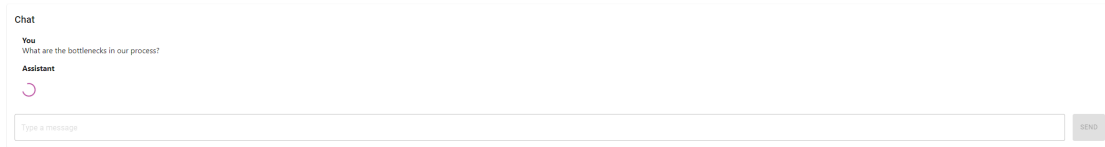


Figure 10. LLM processing indication in the conversational interface.

Every five seconds, the frontend makes a call to the `message_status` endpoint with the `jobid`, `threadid`, and `runid` to check the status of the message processing. Depending on the status returned by the OpenAI system, the frontend takes different actions:

- If the run is **completed**, the message content is retrieved and displayed in the chat interface.
- If the run **fails**, an error code 500 is returned, and the user is informed that there is a problem on the OpenAI API side.
- If the run **requires further action**, the necessary tool is invoked, and the corresponding processing begins.

The user is kept informed of the status of their query within the interface. For instance, while the query is actively being processed by the LLM, the user will see a status message indicating "Working...", as illustrated in Figure 11.

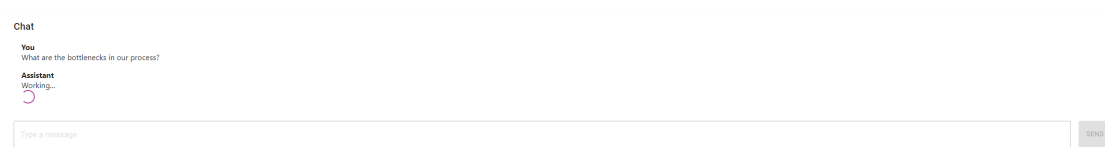


Figure 11. Intermediary processing status in the conversational interface.

Depending on the stage of processing, the chat interface updates to reflect one of several status messages:

- **"Working..."**: Displayed when the LLM is actively processing the current query. This status is accompanied by a visual indicator, such as a spinning icon, signaling that the operation is ongoing. An example of this status is presented in Figure 11.
- **"Retrieving Kronos report"**: Shown when the LLM is querying the database for the results of the Kronos analysis.

- **"Discovering batching strategies"**: Appears when the system is identifying batching strategies within the event log.
- **"Discovering prioritization strategies"**: This message indicates that the LLM is analyzing prioritization strategies used in the event log.
- **"Discovering case attributes"**: When additional attributes of cases within the event log are being examined, this status message is displayed.
- **"Get redesign pattern info"**: Informs the user that the system is gathering detailed information about specific process redesign patterns.
- **"Run has failed, problem on OpenAI API side"**: If the LLM encounters an issue and cannot process the request, this error message is relayed to the user.

These status updates provide users with transparency regarding the progress of their interactions with the LLM, ensuring a clear understanding of the process and timeline for obtaining results.

Once the run is complete and the response is ready, it is conveyed to the frontend and presented to the user, as shown in Figure 12.

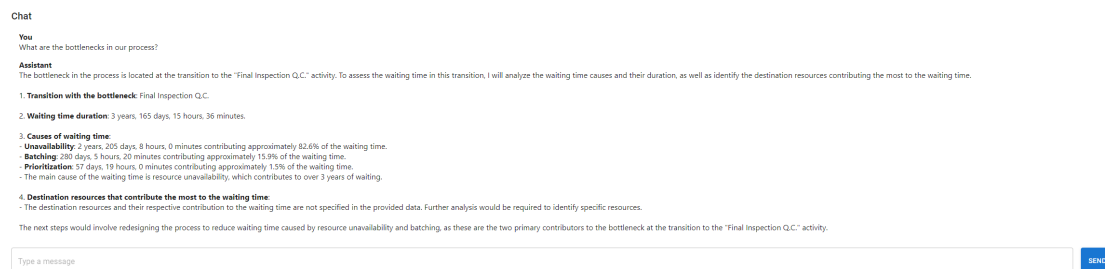


Figure 12. Completed interaction displayed in the conversational interface.

Interaction Tools and Run Status Handling: The backend is equipped with a suite of tools to handle specific actions required by the LLM. Here is a brief overview of possible actions and tools:

The tools available for the backend to utilize are as follows:

As users engage with the conversational interface, the system dynamically adjusts to display the LLM's responses. The chat box expands to accommodate the length of the message output. When users send subsequent messages, they are directed to the `process_request` endpoint, carrying the `threadid` forward. This maintains the context of the conversation, allowing the LLM to recall previous messages within the same thread. Despite this continuity, users have the option to start fresh by simply reloading the page, which resets the conversation thread while preserving the analysis results.

Run Status	Description and Action
Completed	The message is retrieved and displayed to the user.
Failed	An error is indicated, and no further action is taken.
Requires Action	The respective tool is invoked based on the LLM's instruction.
In progress	Run is in progress and Frontend will repeat API call in 5 seconds

Table 1. Run status and corresponding actions in the conversational interface.

Tool	Interface Description
Query Database	Retrieves the Kronos report from the database using an SQL query generated by the LLM.
Discover Batching Strategies	Obtains batching strategies present in the event log.
Discover Prioritization Strategies	Identifies prioritization strategies used within the event log.
Discover Case Attributes	Extracts case attributes, which are additional, non-mandatory columns in the event log that remain constant throughout the case execution.
Get Redesign Pattern Info	Gathers information on specific redesign patterns, including definitions, explanations, and examples.

Table 2. Tools utilized based on the LLM's run status in the conversational interface.

A key feature of the interface is its support for basic markdown, enabling the LLM to enhance message formatting for a better user experience, including headers, bold and italic text, hyperlinks, and line breaks.

This markdown functionality ensures that the chat interface is not only informative but also visually engaging, presenting the content in a structured and easy-to-read format.

By allowing users to communicate through a chat interface and maintaining context throughout their session, the system provides a more natural and engaging user experience. Support for markdown syntax further refines this interaction, presenting information in a clear, formatted manner. The demo showcases the nature of Kronos, illustrating its capacity to evolve and adapt in different scenarios by utilizing conversational interface.

5 Evaluation

5.1 Content Evaluation

5.1.1 Evaluation of Improvement Opportunities Facilitation

To evaluate the efficacy of the conversational interface in facilitating improvement opportunities within process mining, an experimental approach was adopted using four distinct event logs: procure-to-pay, claims management, device repair, and production processes. These logs represent varied business processes, each with its own characteristics and complexities.

The evaluation utilized a comprehensive set of 55 questions tailored to investigate various aspects of waiting times within these logs. Derived from existing research [23], this set of questions was refined to align with the specific aim of identifying improvement opportunities—thus ensuring relevance to the established evaluation goals.

To ensure clarity and directness in the inquiry process, questions were adapted for the experiment’s context. For instance, instead of asking for filtering operations, questions were reframed to elicit direct insights into waiting time issues. This led to a distilled list of 28 questions focused on the discovery of improvement opportunities. An additional 27 questions were created to delve into the specific causes of waiting times as cited in [17].

The experiment was conducted by issuing each question to the LLM through a separate thread, ensuring that the LLM responded without the influence of previous interactions. To facilitate a rapid and structured execution, a script was developed to automate API calls to the application, collecting LLM responses into a consolidated document.

Simultaneously, a manual analysis was conducted to establish a baseline of expected answers, enabling a comparison against the LLM’s responses.

Correctness Assessment The evaluation began with the replication of each event log through the interface, facilitating variability in the LLM’s responses. Cases where the LLM was unable to provide a response were marked, and these instances were assigned a definitive score of 0, signifying an error within the interface’s processing capabilities.

For the remaining LLM responses, a parallel assessment was conducted against the manually formulated baseline answers. This assessment was performed by submitting both the LLM’s response and the baseline answer to a separate OpenAI LLM ChatGPT 4 Turbo instance, which was instructed to evaluate the responses according to a specified scoring guide:

Scoring Guide:

- **0 (No Response/Error):** No response provided or a system error occurred, preventing a response.

- **1 (Unrelated/Error):** Response provided but unrelated or acknowledges an error.
- **2 (Slightly Related):** Mostly unrelated with a vague connection or significant mistakes.
- **3 (Related but Flawed):** Related to the question but with major inaccuracies or very minimal relevant content.
- **4 (Mostly Correct):** Related and mostly accurate with minor errors or slight irrelevance.
- **5 (Fully Correct):** Fully relevant and accurate with no or negligible errors.

The assessing LLM is to strictly provide a single numerical score that reflects the completeness and accuracy of the given answer in relation to the correct answer.

The instructions emphasized the importance of outputting only a single digit to represent the score for each answer's evaluation, thus maintaining the precision of the assessment process. Each query posed to the LLM therefore yielded three key pieces of data: the original LLM's answer, the baseline answer, and the scored evaluation from an independent LLM run.

Following the initial scoring, a random sample of the questions was selected based on the calculated sample size to ensure a rigorous verification process. Simple random sampling, an approach described in [21], that allowed for a detailed manual review of the LLM's answers, comparing them against the baseline to assess their accuracy and completeness.

The first step in the sampling process involved calculating the sample size necessary to achieve a representative and statistically significant subset of responses from the conversational interface. The sample size was calculated based on the following considerations:

- **Population Size (N):** Total number of questions answered by the LLM.
- **Confidence Level (Z):** Standard z-score of 1.96 corresponding to a 95% confidence level.
- **Margin of Error (E):** Desired precision level of 5%.
- **Proportion (p):** Assumed proportion of the characteristic of interest in the population, typically set at 0.5 for maximum sample size.

The sample size was calculated using the formula for a finite population:

$$n = \frac{N \times Z^2 \times p \times (1 - p)}{(E^2 \times (N - 1)) + (Z^2 \times p \times (1 - p))}$$

This formula adjusts the sample size based on the total number of responses, providing a statistically robust estimate of the number of responses needed for manual verification.

Once the sample size was determined, responses were randomly selected from the complete set of LLM outputs to ensure that every response had an equal chance of being included in the review. This random selection process helps to mitigate any bias that might affect the validity of the evaluation.

The selected responses were then manually reviewed to compare their accuracy and completeness against the established baseline answers.

Upon completing the evaluation, a total of 350 questions were processed and analyzed. A statistically representative sample of 183 questions was selected for manual assessment based on the sampling methodology previously described. The results from this manual verification process provided a significant insight into the effectiveness of the LLM in responding accurately to the queries posed.

Error Analysis Of the evaluated responses, a total of 12 were found to result in an error, where the LLM failed to provide a relevant response, corresponding to an error rate of approximately 3.4%. This indicates a relatively low rate of outright failures in the system's operational performance.

The average score for the LLM's responses, excluding those cases where an error occurred (score of 0), was calculated to be 4.3 out of a possible 5. This high average score suggests that, when the LLM provided a response, it was generally accurate and relevant to the queries posed, indicating a strong alignment with the expected answers.

The findings from this evaluation demonstrate that the LLM-based conversational interface performs effectively in identifying and addressing process inefficiencies through its responses. The low error rate coupled with a high average response score underlines the LLM's utility in facilitating improvement opportunities within the scope of process mining. These results confirm the robustness and reliability of the interface, making it a valuable tool for enhancing decision-making in process optimization.

5.1.2 Evaluation of Redesign Suggestions and Their Impact

In alignment with the methodological framework outlined, an evaluative study was conducted to assess the practicality and impact of redesign suggestions offered by the conversational interface. This subsection details the approach taken to evaluate the interface's capability to generate actionable and relevant process improvements.

The interviews, conducted online via Zoom, ranged in duration from 23 to 37 minutes, with an average length of approximately 29 minutes. Each session was recorded using

Open Broadcaster Software (OBS), feedback and scores were documented in real-time to facilitate subsequent evaluation.

Participants were presented with the LLM's analysis of a procure-to-pay process. Through this demonstration, the conversational interface's ability to identify bottlenecks and inefficiencies was highlighted. The aim was to familiarize participants with the interface's analytical capabilities before delving into specific evaluation questions.

Following the demonstration, participants were asked a series of questions designed to gauge the value and applicability of the LLM's responses from a process analyst's perspective. This approach facilitated both quantitative and qualitative feedback, allowing for a nuanced assessment of the interface's utility.

Initial Question The first query posed to the LLM, "Can you identify any bottlenecks or inefficiencies?" served as a primer for the subsequent questions. The intent was not to evaluate the LLM's response but to establish a common ground for understanding the current process's key bottlenecks.

You
Can you identify any bottlenecks or inefficiencies?

Assistant
The analysis of waiting times across various activities indicates several bottlenecks and inefficiencies within the process. Below are the top five activities with the highest total waiting times, along with the biggest contributor to this waiting time in each activity.

1. Activity: Analyze Request for Quotation
 - Total Waiting Time: 7 years, 8 months, 8 days, 1 hour, 5 minutes
 - Waiting Time due to Unavailability: 3 years, 26 days, 6 hours, 36 minutes (39.42%)
 - Waiting Time due to Extraneous Causes: 2 years, 2 months, 1 day, 7 hours, 26 minutes (29.32%)
 - Waiting Time due to Prioritization: 1 year, 2 months, 17 days, 2 hours, 30 minutes (17.22%)
 - Waiting Time due to Batching: 237 days, 12 hours, 30 minutes (10.8%)
 - Waiting Time due to Contention: 89 days, 7 hours, 52 minutes (3.2%)

The main cause of waiting time here is unavailability of resources, contributing to almost 40% of the total waiting time for this activity. Extraneous causes also play a significant role in the delays.

2. Activity: Deliver Goods/Services
 - Total Waiting Time: 1 year, 2 months, 13 days, 17 hours, 20 minutes
 - Waiting Time due to Unavailability: 190 days, 5 hours, 28 minutes (47.54%)
 - Waiting Time due to Contentions: 89 days, 7 hours, 42 minutes (22.6%)
 - Waiting Time due to Batching: 35 days, 9 hours, 42 minutes (8.81%)
 - Waiting Time due to Prioritization: 34 days, 21 hours (8.6%)
 - Waiting Time due to Extraneous Causes: 15 days, 14 hours, 6 minutes (12.43%)

For this activity, resource unavailability is also the major culprit, followed by contention for resources.

3. Activity: Send Invoice
 - Total Waiting Time: 277 days, 13 hours, 44 minutes
 - Waiting Time due to Unavailability: 129 days, 9 hours, 28 minutes (46.46%)
 - Waiting Time due to Contentions: 72 days, 9 hours, 20 minutes (26.27%)
 - Waiting Time due to Extraneous Causes: 37 days, 12 hours, 44 minutes (17.2%)
 - Waiting Time due to Batching: 28 days, 3 hours, 40 minutes (10.07%)
 - Waiting Time due to Prioritization: 14 days, 1 hour, 0 minutes

Send Invoice activity is primarily delayed due to unavailability and contention for resources, signifying resource-related issues.

4. Activity: Confirm Purchase Order
 - Total Waiting Time: 255 days, 19 hours, 48 minutes
 - Waiting Time due to Unavailability: 127 days, 9 hours, 0 minutes (49.96%)
 - Waiting Time due to Extraneous Causes: 35 days, 15 hours, 24 minutes (16.66%)
 - Waiting Time due to Batching: 35 days, 10 hours, 30 minutes (16.52%)
 - Waiting Time due to Contentions: 50 days, 14 hours, 30 minutes (9.92%)
 - Waiting Time due to Prioritization: 10 days, 18 hours, 20 minutes (6.86%)

Figure 13. Partial response from the LLM indicating top bottlenecks

The LLM's response, indicating the top five activities with the highest waiting times and their distribution among various causes, is partially depicted in Figure 13. A manual verification of the provided times confirmed the response's accuracy, ensuring the reliability of the data presented to the participants.

The evaluation proceeded with an inquiry into the LLM's ability to propose specific process redesigns aimed at enhancing throughput based on previously identified

inefficiencies.

Question 2. Based on the inefficiencies identified, what specific process redesigns would you suggest for improving the throughput of process?

LLM's Redesign Proposals The LLM responded with targeted suggestions, recommending *resource scheduling* and the allocation of *extra resources* for the first four activities with substantial waiting times. For the last activity, it advised optimizing *batch and prioritization strategies* to streamline the process further.

The participants assessed the LLM's suggestions for their alignment with the identified bottlenecks. They acknowledged that while most proposals were relevant, the feasibility of batch optimization was questioned, given its minor contribution to overall waiting time.

Each participant rated the LLM's response on a scale from 0 to 5, with 0 indicating no value or relevance, and 5 indicating high value and direct relevance. The scores for the LLM's response to the second question were as follows:

Participant	Value Score	Relatedness Score
1	4	4
2	4	5
3	4.5	3.5
4	3.5	5
5	3.5	3.5

Table 3. Participant scores for the LLM's second response on redesign suggestions

Continuing the evaluation, the third question posed to the LLM aimed to ascertain the expected impact of the suggested redesigns on the overall efficiency of the process.

Question 3. What impact do you predict these suggested redesigns would have on the overall efficiency of process?

LLM's Impact Prediction The LLM projected that the recommended redesigns, particularly those addressing the significant waiting times due to resource unavailability and extraneous factors, would markedly enhance process efficiency. It anticipated that proper resource scheduling and the addition of extra resources could considerably diminish the waiting times for key activities, such as 'Analyzing Request for Quotation' and 'Send Invoice'.

Feedback from participants was unanimously positive regarding the relevance of the LLM's predictions to the event log, reflecting a keen understanding of the process inefficiencies. While most interviewees deemed the response valuable, noting the pragmatic

approach of the suggestions, there was a note about the generality of the predictions. One participant highlighted that without practical implementation and simulation, the true effect of the redesigns could not be accurately gauged.

Participants were asked to rate the value and relatedness of the LLM’s response, with the outcomes shown in Table 4.

Participant	Value Score	Relatedness Score
1	4.5	5
2	4	5
3	5	5
4	3	5
5	4.5	5

Table 4. Participant scores for the LLM’s third response on impact prediction

The responses to the third question underscored the LLM’s ability to provide insightful and related suggestions for process improvement. Despite some concerns about the ability to predict the exact outcomes of the proposed changes, the participants acknowledged the conversational interface’s utility in facilitating a deeper understanding of potential process efficiencies.

The fourth query addressed to the LLM concerned the identification of potential risks and challenges that may arise from implementing the suggested process redesigns.

Question 4. What are the potential risks or challenges associated with implementing the suggested redesigns for process?

LLM’s Response on Risks and Challenges The LLM’s detailed response acknowledged the complexity inherent in process redesign. It outlined potential risks such as misalignment of resource scheduling with actual demand peaks, resistance to schedule changes, the financial burden of additional resources, diminishing returns on investment, and customer dissatisfaction from altered prioritization strategies. Challenges included the complexity of implementing new scheduling, integrating extra resources, optimizing batch sizes, and modifying prioritization strategies without causing operational disruptions.

Participants provided scores reflecting the general applicability of the LLM’s response, with some noting that the information presented was fairly generic and could be applicable to a broad range of scenarios, not specifically tailored to the event log in question. This broader applicability resulted in somewhat lower scores for relatedness and value, as detailed below:

The feedback received underscores the importance of specific, contextual information in evaluating the practicality of redesign suggestions. Despite some participants perceiving the response as relatively basic and widely applicable, the consensus highlighted

Participant	Value Score	Relatedness Score
1	3.8	3.5
2	4	3
3	3.5	3
4	4	3.5
5	4.5	5

Table 5. Participant scores for the LLM’s fourth response on risks and challenges

the need for careful consideration and management of the potential risks and challenges outlined by the LLM.

The fifth question aimed to explore whether the LLM could propose additional strategies for process redesign based on the identified bottlenecks.

Question 5. Is there any other redesign strategies you can offer based on bottlenecks in our process?

LLM’s Additional Redesign Proposals The LLM offered a suite of strategies beyond the initial recommendations, aimed at enhancing process efficiency. These included *fragment elimination* to reduce non-value-adding tasks, *task composition* to combine smaller tasks and reduce transitions, *resequencing* to create a more logical task flow, and *periodic action adjustment* to optimize the timing of recurring actions. The LLM stressed that the applicability of these strategies might vary depending on the specific process characteristics.

Participants provided their evaluations, with some expressing that the suggestions, while broadly applicable, presented a solid foundation for beginning optimization efforts. There was an acknowledgment that the LLM’s recommendations might require follow-up inquiries for clarification or further refinement. The notion of adjusting the periodicity of actions elicited some surprise due to its potentially marginal impact on waiting times. Nonetheless, the overall sentiment was that the responses served as a valuable starting point.

Scores for the LLM’s response to the fifth question were assigned based on the value and relatedness to the event log:

The feedback from the participants at this stage indicates a perception of the LLM’s output as generally applicable yet valuable in the context of the specific process being examined. This reflects the LLM’s potential to act as a preliminary advisor in process enhancement discussions, with its broad recommendations providing a canvas for more detailed, process-specific analysis.

The final question for the LLM focused on proposing a specific resequencing of activities within the event log to address the identified bottlenecks.

Participant	Value Score	Relatedness Score
1	4	4
2	3	3
3	4	3
4	3	3
5	4	4

Table 6. Participant scores for the LLM's fifth response on additional redesign strategies

Question 6. What kind of resequencing you propose in our event log? Give specific example.

LLM's Resequencing Recommendation The LLM recommended several resequencing strategies, particularly focusing on the "Request for Quotation" (RFQ) related activities. It suggested options such as parallel processing of tasks, deferring amendments until after initial analysis, and streamlining the RFQ creation and amendment processes. These recommendations aimed to reduce the considerable waiting times observed between these activities in the event log.

The participants generally found the LLM's response to be highly specific and closely related to the event log's details, reflecting the practical applicability of the suggestions to the given process. While one participant expressed reservations about the LLM's use of total waiting times instead of averages, which could potentially change the perception of bottlenecks, the overall feedback was positive.

The LLM's resequencing proposal was rated by participants for its value and relevance to the event log:

Participant	Value Score	Relatedness Score
1	4	5
2	4	5
3	5	5
4	3.5	5
5	4.5	5

Table 7. Participant scores for the LLM's sixth response on specific resequencing

This series of questions ended with the LLM providing a well-received proposal for resequencing activities. Participants' scores and comments indicate a strong appreciation for the interface's ability to generate specific, actionable insights based on the data within the event log, solidifying its role as a valuable tool in the process analyst's toolkit.

5.1.3 General Assessment of LLM’s Redesign Suggestion Capabilities

To encapsulate the series of evaluations, participants were requested to provide an overarching rating of the LLM’s competency in offering process redesign suggestions and associated information.

The collective ratings from participants reflected a strong confidence in the LLM’s capability to propose meaningful redesign strategies (scores 0-5):

Capability Score
4
4
4
3.5
4

Table 8. Overall participant scores on the LLM’s ability to suggest redesigns

The participants’ assessments, characterized by high scores, outlined the LLM’s utility as a source of redesign recommendations. Furthermore, the scores and feedback from the impact prediction question emphasized the LLM’s adeptness in not only suggesting redesigns but also in providing predictions about their potential impact. These insights highlight the LLM’s potential as a valuable adjunct tool for process analysts in strategizing improvements.

Conclusion The findings from the evaluations conducted through the interviews underscore the LLM interface’s effective role in enhancing the understanding of process inefficiencies. Its proficiency in generating both redesign suggestions and impact predictions illustrates its potential utility in aiding informed decision-making for process optimization. However, the interviews also surfaced some limitations of the LLM, such as occasionally providing general responses that may require further clarification or context-specific tailoring. Additionally, while the LLM demonstrated a strong foundational capability, participants indicated the necessity for follow-up questions in certain scenarios, highlighting the importance of iterative dialogue for refining the LLM’s outputs. These insights are vital for understanding the LLM’s role not as a standalone solution but as a collaborative tool that functions optimally when paired with human expertise.

5.2 Performance Evaluation

The performance of the conversational interface was assessed on various metrics, including response time and response length. During the assessment, the following observations were made:

- The **average response time** for non-zero score responses was recorded at 34 seconds.
- The **average response length** amounted to 1078 characters, indicating an effective rate of nearly 32 characters per second. It is noteworthy to mention that this rate is a simplification, as the system spent portions of the time composing SQL queries or utilizing other tools.

A visual analysis was conducted to study the relationship between total durations and response lengths. Utilizing the matplotlib library in python, a scatter plot was created as shown in Figure 14.

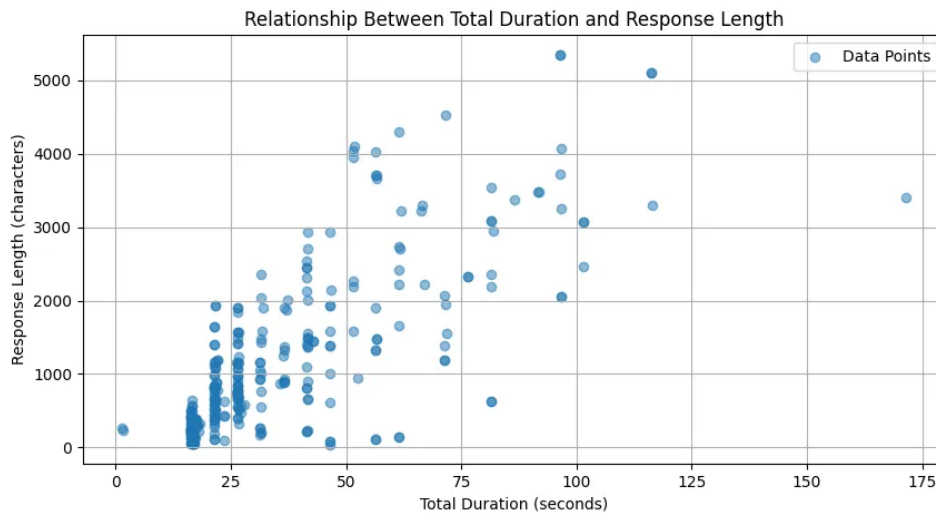


Figure 14. Relationship Between Total Duration and Response Length

A statistical measure of the correlation between these two variables was computed using the numpy library to calculate the Pearson correlation coefficient. The resulting coefficient of 0.77 indicates a strong positive linear relationship. Nonetheless, outliers were observed; particularly, longer responses tended to omit critical details, resulting in lower scores.

The error rate throughout the evaluation stood at 3.4%, which is satisfactorily below the target threshold of 5%. Additionally, the LLM demonstrated the capability to concurrently use multiple tools, such as querying the database while identifying prioritization strategies, showcasing its adeptness at handling complex queries or addressing multiple questions simultaneously.

Another vital aspect considered was the **cost per search**. Although precise cost tracking per search was not feasible due to OpenAI's limitations, the total expense

incurred for testing the dataset with 350 questions over a full day was approximately 20 USD. This translates to an average cost of 18 cents per search, significantly under the maximum allowance of 1 EUR.

Conclusion The performance metrics affirm the LLM’s efficiency and robustness in generating timely and substantial responses. The analysis provides a clear understanding of the LLM’s operational capacity and its financial viability for widespread use, reinforcing the tool’s potential for practical application in process mining tasks.

5.3 Summary of Evaluation Results

This subsection provides a comprehensive summary of all evaluation metrics assessed during the study. The table below aggregates the results from various evaluation aspects, including system performance, correctness of responses, redesign suggestion evaluation, and overall capability scoring by participants.

Performance Scores	
Average Response Time (sec)	34
Average Response Length (characters)	1078
Average Characters per Second	31.7
Error Rate	3.4%
Cost Per Search (USD)	\$0.18
Content Scores	
Average Correctness Score (1-5 scale)	4.3
Redesign Capability Score (0-5 scale)	3.9

Table 9. Consolidated Evaluation Results Summary

The evaluation results demonstrate an efficient performance with an average response time of 34 seconds and a response length of 1078 characters, indicating a robust capability in handling queries effectively. The low error rate of 3.4% further attests to the reliability of the system under test conditions.

The high average correctness score of 4.3 reflects the LLM’s ability to provide relevant and accurate responses when not encountering errors. Participant feedback on redesign suggestions, as detailed in earlier tables, shows a general satisfaction with the LLM’s ability to propose feasible improvements, though the degree of specificity and practical applicability varied among responses.

The cost-effectiveness of the tool was highlighted by an average cost of 18 cents per search, significantly lower than the set threshold, underscoring the LLM’s potential for cost-efficient deployment in real-world scenarios.

6 Discussion, Limitations, and Implications

6.1 Discussion

This subsection summarizes the findings from the evaluation of the LLM interface, emphasizing its facilitation of improvement opportunities, accuracy in technical tasks such as SQL query generation, and the implications of user interaction on the efficacy of the LLM's responses. The discussion aims to outline these elements to derive actionable insights and guide future enhancements.

A significant strength observed in the LLM's performance was its capability to identify and facilitate improvement opportunities within various event logs. The LLM, in most cases, selected appropriate tools for extracting necessary information, demonstrating a high level of decision-making accuracy. This aspect was particularly evident in its flawless execution of SQL queries, which were crafted with precision and aligned well with the data requirements of the tasks at hand. Such capabilities underscore the LLM's potential as a robust tool for process analysis, providing users with reliable data insights to inform process optimization efforts.

The evaluation highlighted the LLM's proficiency in handling technical tasks, especially in writing SQL queries. This skill is critical in scenarios where data retrieval accuracy is paramount. The LLM's ability to generate error-free SQL queries most of the time suggests that it has a solid understanding of database structures and querying languages, which is essential for performing complex data analysis tasks effectively. Enhancing these capabilities could further leverage the LLM's utility in processing and analyzing large datasets, providing deeper insights into process dynamics.

Another crucial observation from the study was the impact of user interaction and query phrasing on the LLM's performance. The way questions were formulated by users significantly affected the LLM's ability to comprehend and respond accurately. Instances where the LLM misunderstood or oversimplified responses were often traced back to ambiguously phrased queries or lack of specific instructions. This finding highlights the importance of developing user guidelines that can assist users in framing their queries more effectively to maximize the utility of the LLM.

While the LLM provided valuable redesign suggestions, the feedback indicated that these were sometimes too generic, which could limit their direct applicability in specific process contexts. Future enhancements should focus on contextualizing the LLM's learning algorithms to adapt its suggestions based on the unique attributes of each process. Tailoring responses to fit the specific operational nuances of different industries or departments could greatly enhance the practical value of the LLM's recommendations.

6.2 Implications for Research

This subsection explores the broader academic implications arising from the current study of the LLM’s application in process optimization.

The findings from this study on the application of Language Model-based (LLM) technologies in process optimization reveal significant research opportunities and underscore existing gaps. The research highlighted the need for a deeper understanding of how LLMs process and respond to complex queries within varied contexts. Future research should aim to unravel the decision-making mechanisms of LLMs to enhance model transparency and explainability, which are critical for their application in industries.

There is also a critical need to tailor LLMs to specific industry domains. We noticed that while the LLM performed well in general process analysis, it often required significant user input for context-specific responses. Investigating adaptive learning strategies that enable LLMs to continually learn from domain-specific data and interactions could lead to more personalized and effective AI tools. Such strategies may involve hybrid models that combine domain-specific knowledge with the LLM’s broad capabilities, possibly through techniques like transfer learning or model fine-tuning.

Moreover, this research showed the lack of standardized, robust metrics to evaluate LLM performance across various applications, as own metrics were used for evaluation. Developing a set of quantitative and qualitative metrics could help in accurately measuring the efficacy and efficiency of LLM responses, serving as benchmarks for performance across different settings.

Lastly, the appropriate balance between automation provided by LLMs and necessary human oversight was identified due to possibility of the LLM to generate incorrect responses sometimes. Research should explore how best to integrate LLM recommendations with human decision-making to optimize both the speed and accuracy of process improvements, determining when automated suggestions should be implemented directly and when they require human validation.

6.3 Implications for Industry

This subsection discusses the practical implications for industry stemming from the findings of this study on the LLM’s capabilities in process management.

The practical implications for the industry from this study’s exploration of LLM capabilities in process management highlight several key considerations. Because LLMs are not ideal and may produce incorrect information sometimes, LLMs should not be seen as replacements but as enhancements to existing workflows, capable of analyzing extensive datasets and aiding in process improvements. Industries should carefully assess where LLMs can be integrated into their current operations to boost productivity without disrupting established practices.

The effectiveness of LLMs greatly depends on their customization and training to

meet specific industry demands. Given the diversity in industry processes and requirements, LLMs must be fine-tuned using sector-specific data and adjusted to fit particular operational terminologies and protocols.

Finally, clear communication is vital for the successful adoption of LLMs. The study underscored that the quality of user queries significantly impacts LLM performance, indicating the need for adequate training for employees on how to effectively formulate queries.

6.4 Limitations

This subsection addresses the limitations encountered during the study, which are crucial for interpreting the findings and recommendations with an appropriate level of caution. These limitations also suggest areas for further refinement in future research and application of the LLM in process management.

The limitations of this study largely stem from its methodological setup and affect how the results can be applied more broadly. The results' applicability is restricted mainly to the specific scenarios tested: procure-to-pay, claims management, device repair, and production event logs. This means the findings might not directly apply to different contexts or datasets not examined in this study.

Additionally, the quality and scope of the input data significantly influenced the LLM's effectiveness. The study relied on detailed event logs from specific business processes. If the data were incomplete, outdated, or biased, the LLM's responses could be less accurate and helpful.

Another noted limitation was the LLM's occasional delivery of general or unclear responses. These responses often needed extra interpretation, showing that the LLM sometimes struggles to provide detailed, context-specific advice without additional input. This issue is particularly critical in complex scenarios where detailed and specific guidance is crucial.

These methodological limitations suggest that the study's conclusions should be interpreted with caution, particularly when considering their application beyond the specific scenarios tested.

7 Conclusion

This thesis explored the integration of a Language Model-based (LLM) conversational interface into the Kronos process mining tool, aimed at enhancing its usability and functional capabilities. The development followed a structured methodology that incorporated requirements gathering, iterative testing, and rigorous evaluation of the interface's performance and utility.

The results demonstrate that the conversational interface improves the process of identifying and suggesting improvements within business process management by leveraging LLM capabilities. The system effectively facilitated improvement opportunities, generated accurate SQL queries for data retrieval, and provided practical redesign suggestions. Error rates were low, and the response metrics indicated a high degree of efficiency and user satisfaction. By integrating this interface, Kronos now supports a more interactive environment where users can receive responses and actionable insights, effectively transforming static data presentation into a more engaging analytical dialogue. This advancement in Kronos's functionality empowers users to make informed decisions faster and more accurately, leveraging the sophisticated natural language processing capabilities of the LLM.

Future developments could focus on expanding the LLM's contextual understanding and its ability to generate more customized responses based on specific industry needs. Further research might also explore the integration of additional AI capabilities to enhance decision-making processes and automate more complex process management tasks.

The findings from this study contribute to the broader field of AI in business process management by demonstrating the potential of conversational interfaces to enhance analytical tools and decision-making processes.

References

- [1] Mahnaz Sadat Qafari A. Berti. Leveraging large language models (llms) for process mining (technical report). Technical report, arXiv.org, 2023.
- [2] Wil M.P. van der Aalst A. Berti, Daniel Schuster. Abstractions, scenarios, and prompt definitions for process mining with llms: A case study. *arXiv.org*, 2023.
- [3] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623, 2021.
- [4] Alessandro Berti and Mahnaz Sadat Qafari. Leveraging large language models (llms) for process mining. 2023.
- [5] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [6] Kiran Busch, Alexander Rochlitz, Diana Sola, and Henrik Leopold. Just tell me: Prompt engineering in business process management. 2023.
- [7] Mike Cohn. *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional, 2004.
- [8] Mike Cohn. *Succeeding with Agile: Software Development Using Scrum*. Addison-Wesley Professional, 2009.
- [9] John W. Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. SAGE Publications, 2014.
- [10] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30(4):681–694, 2020.
- [11] Yago Fontenla-Seco, Sarah Winkler, Alessandro Gianola, Marco Montali, Manuel Lama, and Alberto Bugarín-Diz. The droid you’re looking for: C-4pm, a conversational agent for declarative process mining . 09 2023.
- [12] Michael Grohs, Luka Abb, Nourhan Elsayed, and Jana-Rebecca Rehse. Large language models can accomplish business process management tasks, 2023. <https://arxiv.labs.arxiv.org/html/2307.09923>.

- [13] Michael Grohs, Luka Abb, Nourhan Elsayed, and Jana-Rebecca Rehse. Large language models can accomplish business process management tasks. 2023.
- [14] Karen Hao. The training of ai models is outsourced to low-paid workers. *MIT Technology Review*, 2019. <https://www.technologyreview.com/2019/10/25/132244/ai-training-new-york-times-kenya-openai-gpt-2/>.
- [15] Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, 2004.
- [16] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.
- [17] Katsiaryna Lashkevich, Fredrik Milani, David Chapela-Campa, Ihar Suvorau, and Marlon Dumas. Why am i waiting? data-driven analysis of waiting times in business processes. In Marta Indulska, Iris Reinhartz-Berger, Carlos Cetina, and Oscar Pastor, editors, *Advanced Information Systems Engineering*, pages 174–190, Cham, 2023. Springer Nature Switzerland.
- [18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [19] Heidy M. Marin-Castro and Edgar Tello-Leal. Event log preprocessing for process mining: A review. *Applied Sciences*, 11(22):10556, 2021.
- [20] Fredrik Milani, Katsiaryna Lashkevich, Fabrizio Maria Maggi, and Chiara Di Francescomarino. Process mining: a guide for practitioners. In *Proc. of RCIS*, pages 265–282. Springer, 2022.
- [21] Shagofah Noor, Omid Tajik, and Jawad Golzar. Simple random sampling. 1:78–82, 12 2022.
- [22] Tom Smith. On location at celonis world tour: Large language models and object-centric process mining, 2023. <https://accelerationeconomy.com/ai/on-location-at-celonis-world-tour-large-language-models-and-object-centric-process-mining/>.
- [23] Dirk Fahland U. Jessen, Michal Sroka. Chit-chat or deep talk: Prompt engineering for process mining. *arXiv.org*, 2023.
- [24] W. M. van der Aalst. *Process mining: discovery, conformance, and enhancement of business processes*. Springer, 2011.
- [25] Wil M. P. van der Aalst. *Process Mining: Data Science in Action*. 2nd edition, 2016.

- [26] B.F. van Dongen, R.S. Mans, and W.M.P. van der Aalst. The prom framework: A new era in process mining tool support. In *Applications and Theory of Petri Nets*. Springer, 2005.
- [27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, 2017.
- [28] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.
- [29] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface’s transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.
- [30] Robert K. Wysocki. *Effective Project Management: Traditional, Agile, Extreme*. Wiley, 6th edition, 2011.
- [31] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213. Association for Computational Linguistics, 2018.

Appendix

I. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Maksym Avramenko**,
(author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

LLM-based Interface for Data-Driven Waiting Time Analysis of Business Processes,

(title of thesis)

supervised by Fredrik Payman Milani and Katsiaryna Lashkevich.

(supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.
3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.
4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Maksym Avramenko

14/05/2024