AI-Assisted Automation for Maintaining User Documentation

University of Turku
Department of Computing
Master of Science (Tech) Thesis
Software Engineering
May 2026
Joni Lassila

UNIVERSITY OF TURKU
Department of Computing

Joni Lassila: AI-Assisted Automation for Maintaining User Documentation

Master of Science (Tech) Thesis, 52 p.
Software Engineering
May 2026

This study investigates how artificial intelligence can support the process of updating user documentation in an enterprise software environment, where documentation maintenance has traditionally been a manual and time-consuming task.
The research was conducted as a case study in an enterprise environment, where an AI-assisted solution was designed and implemented using Microsoft Power Platform technologies. The solution uses large language models to analyze software change descriptions, such as release notes, and compares them against existing documentation to identify outdated content. The process follows a human-in-the-loop approach, in which the documentation maintainer reviews and approves all changes before they are applied.
The evaluation was performed using real documentation update cases. The results show that the AI-assisted approach reduces the time required for documentation updates by approximately 48%, particularly in identifying relevant documents and sections. The solution performed best when modifying existing documentation, while generating new content remained more challenging. The findings highlight the importance of human control to ensure accuracy and consistency with documentation standards.
The study concludes that AI-assisted solutions effectively support documentation maintenance by reducing manual effort and improving efficiency, while still requiring human validation. The results suggest that such systems function most effectively as decision-support tools rather than fully autonomous solutions.

Keywords: artificial intelligence, large language models, intelligent document processing, documentation maintenance, Microsoft, Microsoft Copilot Studio Agent

Contents

List of Figures

List of Tables

1 Introduction

Software documentation plays a crucial role in ensuring that software is maintainable and understandable. In enterprise environments, user documentation is continuously updated alongside the software, introducing challenges in maintaining accurate and up-to-date documentation. [1], [2]

Updating user documentation is often a manual process in which the documentation maintainers analyze software changes, identify affected documentation, and update it accordingly. This process can be time-consuming and error-prone, especially when the software has a large number of documents that consist of multiple related documents. Additionally, as software development follows fast-paced practices, such as continuous integration and delivery, maintaining up-to-date and accurate documentation becomes more challenging.

The usage of artificial intelligence (AI) in enterprise environments has increased significantly in recent years. Organizations are increasingly integrating AI into their business processes to improve efficiency and automate tasks of various complexities. As shown in Figure 1.1, the use of AI has grown steadily over time, with a recent increase in the usage of generative AI. This trend is also reflected in recent surveys, where most organizations report using AI in at least one business function. [3]

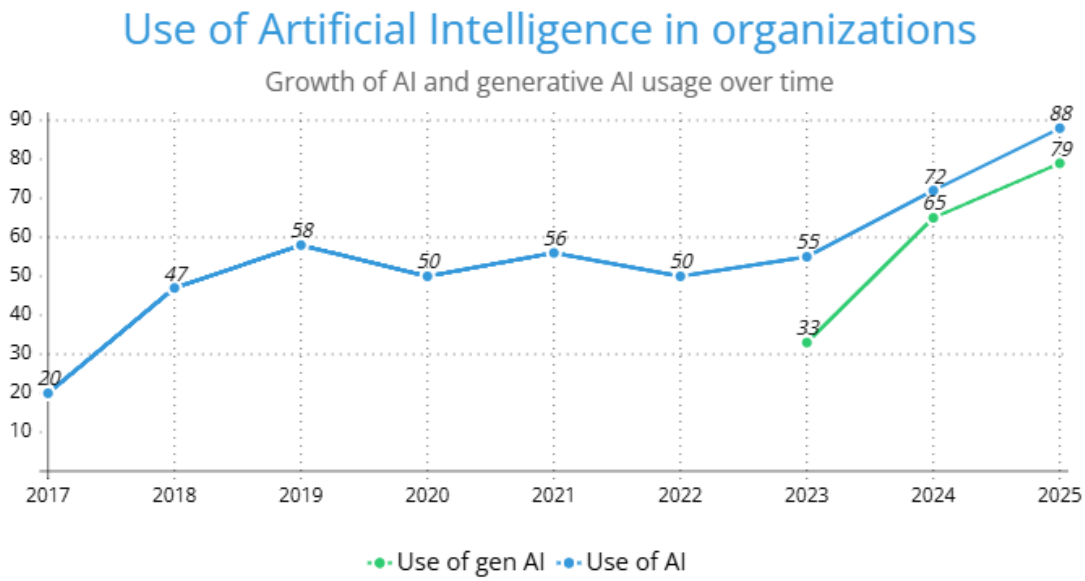

Figure 1.1: Illustrative trend of increasing AI usage in organizations [3].

Advances in artificial intelligence, such as large language models, have introduced new opportunities for supporting these tasks. These models have demonstrated advanced capabilities in both understanding and generating natural language, making them highly effective for tasks such as text analysis, summarization, and content generation [4]. These capabilities suggest that the models can be utilized for maintaining software documentation within an organization by assisting in identifying outdated documentation and generating update proposals based on software changes.

The objective of this thesis is to investigate how AI can support the process of updating user documentation. The study focuses on evaluating how effectively and accurately language models can identify outdated documentation and assist in implementing updates based on software change descriptions.

To achieve this objective, an AI-assisted solution was designed and implemented using Microsoft Power Platform technologies [5]. The solution utilizes multiple components, including conversational interfaces, workflows, and language model-based

analysis to analyze software change descriptions, such as release notes, and compare them with existing documents to identify affected documentation and generate update proposals. The solution utilizes a human-in-the-loop approach, where the documentation maintainer reviews and approves the proposed updates before they are applied.

1.1 Research questions

The objective of this thesis is to evaluate the role of artificial intelligence in supporting user documentation maintenance in enterprise environments. Based on this objective, the study is guided by the following research questions:

1. How does artificial intelligence-assisted automation change the process of updating user documentation compared to manual methods?

2. How can language models be used effectively to support the identification and implementation of user documentation?

3. What are the advantages and limitations artificial intelligence and automation bring to the maintainability of user documentation?

1.2 Thesis structure

This thesis is structured as follows. Chapter 2 presents the theoretical background and related research, including large language models, user documentation, and artificial intelligence in document processing. Chapter 3 describes the current documentation process and its challenges in an enterprise environment. The chapter introduces the manual workflow and highlights the limitations that motivate the need for automation. Chapter 4 introduces the design and implementation of the AI-assisted solution. It describes the solution architecture, key components, and

details of the developed solution. Chapter 5 presents the empirical evaluation of the solution. The collected data of the solution is analyzed to evaluate the effectiveness of the AI-assisted approach in supporting documentation maintenance. Finally, Chapter 6 discusses the results and presents the future improvements, and Chapter 7 concludes the thesis by summarizing the key contributions.

1.3 Declaration of Generative AI use

This thesis employed generative AI-based tools only for auxiliary tasks in accordance with the University of Turku's guidelines. Grammarly was used for grammar checking, spelling correction, and improving the clarity of the text. ChatGPT was used to support the planning of the thesis structure.

All content was reviewed and verified by the author, and the research design, implementation, analysis, and conclusions were produced without reliance on AI-generated results.

2 Theoretical background and related research

This chapter presents the theoretical background and related research on the artificial intelligence-assisted documentation update solution developed in this thesis. It introduces the key concepts and technologies that form the foundation of the implemented solution, including large language models (LLMs), user documentation in software development, intelligent document processing, and AI-based tools for document automation.

The chapter first discusses LLMs and their ability to process natural language and identify semantic relationships in text. These capabilities are relevant for this study, as the implemented solution relies on language models to analyze documentation and detect outdated information based on software change descriptions.

Next, the chapter examines the role of user manual documentation in software development and how AI can be applied to document processing tasks. Intelligent document processing enables automated analysis and interpretation of textual data, providing the basis for automating parts of the documentation maintenance process.

Finally, the chapter introduces AI-based tools used for document automation. These technologies integrate language models and workflow automation to support document analysis, content generation, and enterprise process integration, forming the foundation for the AI-assisted documentation update solution implemented in

this study.

2.1 Large Language Models and semantic understanding

LLMs are large-scale, pre-trained statistical language models based on neural networks, which are capable of processing and generating natural language. The models learn patterns in language by predicting the probability of the next token based on the given context, which allows them to generate contextually relevant text. [6]

LLMs typically refer to transformer language models with hundreds of billions of parameters, trained on massive amounts of textual data, enabling them to capture complex semantic relationships and perform tasks such as text generation, question answering, translation, summarization, and even programming assistance. [4]

These models are typically based on the Transformer architecture introduced by Vaswani et al. [7], which relies on attention mechanisms to model relationships between tokens in a sequence.

The success of LLMs has brought out skepticism about their ability to capture meaning or human concepts. Research by Piantadosi [8] argues that LLMs capture key aspects of meaning, which is still imperfect, while mirroring meaning in human cognitive theories and philosophical approaches to language. This behavior arises from the training objective of language models, which is to predict the next token based on context. However, recent studies suggest that LLMs may struggle with fine-grained semantic understanding and may rely on pattern recognition rather than true comprehension [9].

LLMs have been successfully applied to a wide array of tasks, including text generation, summarization, translation, question answering, and code generation [4]. LLMs have also been recently used as the core component of AI agents, which

enable interaction with external tools, retrieve information, and perform tasks in dynamic environments. [10]

LLMs have demonstrated remarkable capabilities, but they also contain several limitations and risks that need to be considered. According to Weidinger et al. [11], these risks can be categorized into six areas. First, models can reproduce discrimination, exclusion, and hate speech, which mirrors harmful stereotypes and unequal performance across languages and social groups. Second, they may have information hazards, causing sensitive data leakage if such data is present in training data. The third risk is that models can hallucinate or generate misinformation, which results in an output appearing confident and correct but is incorrect or misleading, since models are trained to predict the next word based on probability, but this doesn't guarantee that the sentence is factually correct. Fourth, models can be used maliciously to spread targeted disinformation, malware, or fraud. Fifth, the language models may cause harm in human-computer interaction. Due to their conversational nature, users may view language models as human-like, even though they don't have understanding or intent. This may lead users to trust its outputs more and reveal sensitive information that they would not otherwise share. The conversational nature of language models may enable manipulation, as users may respond to the system as if it were a real human rather than a computer. Additionally, conversational systems may promote harmful stereotypes. For example, assigning gendered names to assistant systems may associate roles, such as assistants, with specific genders. Finally, large-scale models may also introduce environmental and socioeconomic harms. The training and operation of language models require a large amount of computational resources, which can lead to increased energy consumption and environmental impact. The benefits of these technologies may not be evenly distributed, increasing inequality and affecting job markets by automating tasks that were previously performed by humans.

Language models provide significant benefits, but their limitations highlight that their outputs cannot be assumed to be fully risk-free or reliable. Their use in enterprise environments requires careful consideration, safeguards, and human supervision.

2.2 User manual documentation in software development

User documentation has an essential role in ensuring that enterprise software and services are used efficiently and effectively. Well-written manuals allow internal and external users, such as help desk specialists and end customers, to understand the software's functionalities and troubleshoot issues. Large enterprises' user manuals often cover hundreds of functionalities of varying complexity, and maintaining them is an essential part of the product's life cycle. [2]

The IEEE standard 1063-2001 defines software user documentation as "guides users in installing, operating, and managing software of any size, and conducting those aspects of software maintenance that do not involve modification of the software source code" [2]. The documentation should consider the audience, purpose, and form. The audience determines the level of technical details to be used in the documentation. User documentation in enterprises often provides knowledge to multiple user groups, from software administrators to end users with limited technical knowledge. Therefore, a single version of documentation written from a single perspective is ineffective at guiding all user groups. The purpose defines the documentation's focus, whether it guides users through software workflows, explains software functionality, or helps troubleshoot errors. The form refers to the way documentation is delivered to the user, such as manuals or videos. Different forms provide different levels of interaction and accessibility depending on the user's needs.

[2]

Software documentation is not created at any specific phase but throughout the entire software life cycle. The IEEE 15289 international standard specifies that documentation is developed and revised during all stages of the software and system life cycles. This standard has been created to keep documentation up to date, as studies have shown that 40% to 60% of software maintenance time is spent on understanding the software due to outdated or insufficient documentation. [2]

This highlights the importance of maintaining accurate and up-to-date documentation as part of the software development process. With the rise in popularity of continuous integration and continuous delivery, documentation can be kept up to date alongside the software development.[2]

2.3 Artificial intelligence in document processing

As the software in enterprises grows in scale and complexity, the documentation increases accordingly. Maintaining large-scale documentation manually is time-consuming and error-prone, and documentation often becomes outdated as software evolves [12]. This has led to a demand for automated document processing using AI-based solutions that can update documentation more efficiently and accurately.

Luca [13] defines document analysis as the process of transforming unstructured text into structured information that can be used for automation and decision-making. Previous studies have shown that AI-assisted documentation systems can improve efficiency. For example, Ju [14] reports that, in a healthcare context, a generative AI-based documentation system reduced documentation time by approximately 40% compared to traditional methods.

Natural language processing (NLP) has appeared as a transformative technology for enhancing and automating document analysis. NLP allows the extraction of insights from text documents such as user manuals. This plays a critical role in

enabling machines to interpret, understand, and generate meaningful and helpful human language. NLP includes several core concepts. Tokenization splits text into smaller parts, such as words or sentences, enabling efficient processing [15]. Named Entity Recognition identifies elements such as dates and names within the document. Part-of-speech tagging supports text analysis by identifying nouns and verbs, enabling NLP systems to comprehend sentence structure. Sentiment analysis determines the tone of the text and categorizes it as neutral, negative, or positive [13]. Text classification categorizes the text into predefined labels or categories, which can be used in topic classification or spam detection. Text summarization condenses a large amount of text while preserving key information. [13] These techniques form the core of AI-driven document processing, enabling systems to identify outdated content, propose updates, and maintain consistency across enterprise documentation.

Recent advances in deep learning have significantly improved NLP by developing transformer models such as BERT [16] and GPT [17]. Transformers are deep learning models that use self-attention mechanisms to better understand the relationships between words in a sentence [13].

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained transformer-based model developed by Google. The model uses a bidirectional transformer architecture that learns relationships between two words in both left and right directions, unlike earlier models that processed text in a single direction. The model uses two objectives to pre-train: masked language modeling (MLM), where it predicts a masked random word using only context, and next sentence prediction (NSP), where it learns the sentence relationships by predicting from two pairs of sentences whether the second sentence logically follows the first one or is unrelated. These pre-training techniques enable BERT to achieve state-of-the-art results in NLP tasks such as question answering, sentiment analysis, and named entity recognition. [16]

GPT (Generative Pre-trained Transformer) was developed by OpenAI [17] and is based on a generative pre-training approach, in which a language model is first trained on large amounts of unlabeled text data. The model learns to predict the next word in a sequence based on the given context, enabling it to capture patterns and long-range dependencies in text.

The training of GPT consists of two stages. First, the model is pre-trained on a large corpus of text using unsupervised objectives to learn general language representations. Next, the model is fine-tuned on specific target tasks using the corresponding supervised objective. [17]

The results, presented in the study, demonstrate that generative pre-training combined with fine-tuning improves the performance across a wide range of tasks, including question answering, text classification, and semantic similarity. [17]

2.4 Artificial Intelligence Tools for Document Automation

Organizations are producing large volumes of semi-structured and unstructured documents daily. Managing, analyzing, and extracting value from these documents has become a challenge. In response to this, AI has become increasingly popular and is now an important part of automating various workflows. Automating workflows helps to reduce manual effort and improve the consistency and quality of documents. Leveraging AI-based tools enables organizations to update, generate, and manage documentation more dynamically, even in fast-growing software and agile methodologies. [18]

Intelligent Document Processing (IDP) is an AI technology used to automate the extraction and processing of data from documents. It consists of multiple stages, including document classification, data extraction, validation, enrichment, and inte-

gration into downstream systems. These stages shape a pipeline where documents are first analyzed and interpreted, then relevant information is extracted, validated, and finally transformed into a usable format. [18]

Traditional Optical Character Recognition (OCR) focuses on converting documents into readable text. IDP extends this process by integrating AI techniques such as Natural Language Processing (NLP) and Machine Learning (ML). These technologies enable the interpretation, classification, and extraction of information from documents [19]. This is important in enterprise environments where the majority of data exists in unstructured formats that are difficult to process using traditional approaches [20].

Mandvikar [18] highlights that integrating LLMs into IDP workflows enhances semantic understanding and contextual interpretation of document content. LLMs can improve tasks such as document classification, extraction, and validation by analyzing the meaning of text instead of keyword-based approaches. For example, LLMs enable the identification of relationships within documents, the interpretation of information, and the generation of structured outputs from unstructured input.

This capability is relevant in document-intensive environments, where large volumes of semi-structured documentation require analysis and updates. The implemented solution in this thesis can be seen as an extension of traditional IDP approaches, where instead of extracting data, the system identifies outdated documentation content and generates update proposals based on software change descriptions.

Achieving such automation requires practical development tools such as Microsoft Copilot Studio [21]. Microsoft Copilot Studio is a low-code platform that enables organizations to create copilots for specific functions to help users gather knowledge and create custom virtual assistants that combine LLMs, generative AI, and rule-based logic. The platform integrates with the Microsoft Power Platform,

allowing agents to connect to Power Apps, Power Automate, and AI Builder. Power Apps allows organizations to publish their agents across different channels, such as Microsoft Teams, website integration, or other standardized enterprise communication channels, which enables better accessibility. Power Automate allows for interaction with enterprise systems through automated workflows, allowing tasks to be executed without manual intervention. AI Builder enables building, training, and publishing custom AI models, including GPT-based models that can understand natural language, summarize text, and generate new content. [21]

Additionally, several other similar IDP platforms exist for building AI-assisted agents. These include IBM watsonx Assistant, Amazon Lex, and Google Vertex AI. Microsoft Copilot Studio was selected for this solution due to its seamless integration with the Microsoft Power Platform ecosystem, making it suitable for an enterprise where Microsoft technologies are already widely used. Additionally, the support for human-in-the-loop interactions was a key requirement in the documentation maintenance process.

Combining these components enables the development of the solution presented in this thesis, which supports documentation maintenance through AI-assisted analysis and update generation.

3 Current State of Manual Processing

The previous chapters examined the theoretical background of software documentation and artificial intelligence-based methods. The purpose of this chapter is to describe how user manuals are currently produced, updated, and managed in practice. The beginning of the chapter outlines the existing manual workflow and how user documentation is currently managed in practice. After the current state is discussed, the key challenges and limitations of the manual process are introduced. Finally, the research focus of the case study is introduced, and the data sources used to analyze the current documentation process.

3.1 Current process for updating manuals

The current process for updating user manuals for the software has primarily been manual and involves multiple steps before the manual is updated for the customers. This process relies on a documentation maintainer who transforms technical changes into user-friendly content. When a new software change is introduced, the development team gathers a list of functional changes in the release note. This release note is reviewed by the documentation maintainer, who identifies all necessary changes in existing manuals and manually edits them. This update includes finding relevant documents and sections affected, rewriting instructions, and ensuring consistency of

the manuals. These documents are used both internally and externally, which the maintainer must consider when ensuring that the content is clear enough for end users and technically accurate for customer support.

Figure 3.1 presents the manual workflow used to update user documentation. The developers write release notes describing the software changes. After the release notes are published, the documentation maintainer reviews the changes to determine which parts of the documentation are affected. The maintainer then searches the documentation and identifies the sections that require modification. The documentation is edited manually and saved after each update. If multiple documents are affected by the software change, the maintainer must repeat the process until all necessary updates have been applied.

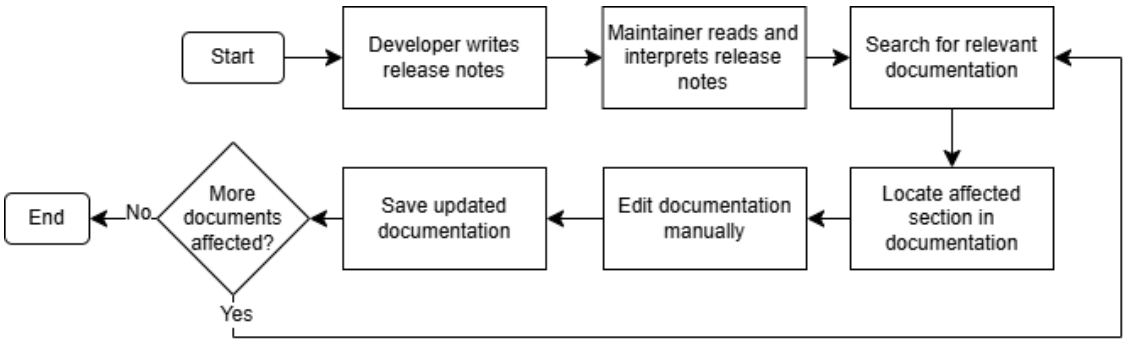

Figure 3.1: Flowchart of the manual update process.

3.2 Challenges and limitations

The process of manually updating documentation, as described in the previous section, contains several challenges and limitations that make it difficult to maintain up-to-date documentation in a rapidly changing software environment.

The primary challenge is the time-consuming aspect of manual editing. Updating the documentation requires the maintainer to read through the release notes,

understand the changes, and find all locations in the existing manuals where modifications are needed. Even the smallest software functionality changes can have a significant impact on multiple documents, leading to a considerable workload. Because the software documentation often relies on a single documentation maintainer, their availability becomes a bottleneck. Documentation maintainers often have several responsibilities beyond documentation tasks, which may delay documentation updates during periods of high workload. This leads to user manuals not being fully up-to-date.

Another challenge is the risk of human error. Since the process relies heavily on manual rewriting, inconsistencies can be introduced unintentionally. These inconsistencies might go unnoticed until an end-user encounters a problem, which increases customer support requests and reduces trust in the documentation. Without an automation tool, verifying the consistency of every update in manuals requires significant manual effort.

The manual process struggles to keep pace with the continuous development as described earlier. Modern agile methodologies can introduce functionality fixes or feature changes more frequently. When documentation updates depend only on manual rewriting, it becomes difficult to match the pace of updates. This leads to situations where a new or updated feature is available to users before the documentation is updated accordingly, leading to reduced usability and increased support contacts.

These challenges highlight why manual documentation maintenance becomes difficult to scale. They provide an important reason for exploring AI-assisted solutions to reduce manual effort and ensure documentation remains up-to-date with software updates.

3.3 Case Description

The case study examines how artificial intelligence can support the process of updating user documentation. The empirical part of the thesis focuses on providing answers to research questions introduced in Chapter 1 by analyzing how AI-based automation affects the documentation updating process, how effectively language models identify outdated content, and what kind of advantages and limitations arise when automation is introduced to real documentation.

The study utilizes data from the software's existing documentation system. The primary data sources are earlier versions of the user manuals and release notes. The existing manuals serve as a baseline for the AI to analyze for potential outdated content. Release notes serve as the primary source of information for software changes.

The evaluation is based on a set of documentation update cases from real documentation maintenance scenarios. In each case, the documentation update is performed using both a manual approach and the implemented AI-assisted solution. This approach enables a direct comparison between traditional manual documentation practices and AI-assisted workflow.

The collected results are used to evaluate the effectiveness of the solution in terms of time efficiency, correctness, and the amount of manual effort required. These results provide useful insights into the practical applicability and limitations of AI-assisted documentation maintenance in an enterprise environment.

4 Implementation of the solution

This chapter presents the implementation of an AI-assisted tool developed to support and partially automate the process of updating user documentation, presented in Chapter 3. It explains how the solution was designed, implemented, and integrated into the existing documentation maintenance flow.

The focus is on the technical implementation of the solution and decisions made during its development. The chapter first presents a high-level overview of the solution to present its main objective, followed by the architecture of the solution and the interaction between its key components. Next, the technologies used in the solution and their roles are discussed. Finally, the implementation stages are presented.

4.1 Overview of the solution

The implemented solution supports and partially automates the process of updating user manuals by using artificial intelligence-based tools. The solution's aim was not to fully replace manual documentation, but rather it is designed to assist the documentation maintainer by finding sections of existing manuals that require updates and generating proposed changes based on software changes. The overall objective is to help keep the documentation up to date by reducing the manual effort.

At a high-level, the solution operates as a pipeline that processes the existing user documentation with release notes describing the software updates. The pipeline

analyses these inputs to detect outdated content and inconsistencies. Based on this analysis, update suggestions are generated to reflect the latest software version while maintaining the original documentation structure and terminology.

Interaction with the solution is handled through an interface implemented using Microsoft Copilot Studio. The interface serves as an entry point to the solution by allowing the document maintainer to input release notes or changes in a free-text format for analysis and receive proposed updates. Once the solution is initiated, a workflow is invoked that manages data preparation, AI-based text processing, and the storage of results for review.

The automation is implemented using Power Automate flows, which handle different stages of the process. These flows handle document retrieval, preprocessing, and interaction with AI Builder models for analysis. During the analysis, the flow prepares the documentation into a suitable text format and uses AI Builder prompts to identify outdated or missing information based on the change description.

The solution is designed to function as a support tool rather than a fully autonomous system. Human contribution remains an essential part of the process, ensuring the generated content is accurate, appropriate, and meets the organizational documentation standards. This design was aligned with the requirements of the enterprise documentation maintenance team and addresses known limitations of AI-based tools, such as infrequent inaccuracies or hallucinations. Limitations and challenges are discussed further in Chapter 6.3.

4.2 Architecture of the solution

This section describes the architecture of the AI-based solution and explains how the main components interact to enable the automated documentation update process. Figure 4.1 illustrates the high-level architecture and the flow between components.

The solution starts when the document maintainer interacts through the con-

versational interface. The agent collects a natural language description of software changes, for example, release notes or free-form text, and triggers the workflow. After collecting the description of software changes, the agent presents the analysis results to the user and waits for the human approval step before modifying any documentation, enabling a controlled approach.

The automation is implemented using two workflows that separate the analysis from the document modification. The first workflow is an analysis flow, which is responsible for retrieving documentation, converting documents into text format, and proposing update requests. The analysis flow begins by accessing the SharePoint document library and retrieving a list of user manual documents. These documents are stored in a Word format, and the workflow utilizes an external document processing service to convert the document content into plain text for analysis using AI. After preprocessing the document files, each file goes through an AI Builder prompt, which performs an analysis. This analysis compares the provided change description with the existing documentation content to identify outdated or missing information. This outputs a change proposal, including the original text segment and the suggested replacement. The proposals are returned to the Copilot Studio agent, which then presents them to the user for review.

The second workflow is a document update flow, which is triggered only after the document maintainer approves the proposed changes. The flow retrieves the document that needs to be updated from SharePoint and applies the approved modifications using an external API action for text replacement. To reduce the risk of unintended changes and ensure reversibility, the flow creates a backup of the original file before overwriting it with the updated version. After the update has been applied, the flow returns the status summary to the user and offers the option to process additional changes.

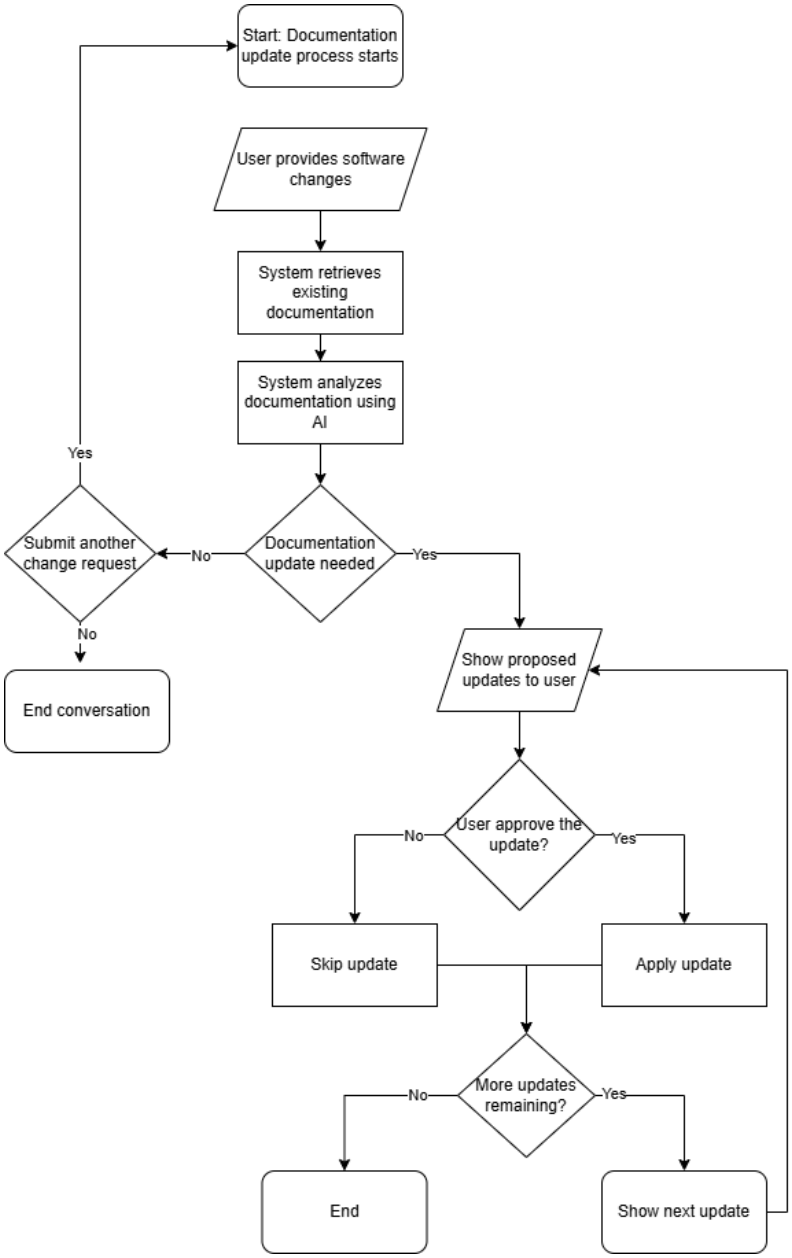

Figure 4.1: High-level flowchart of the implemented solution, illustrating the interaction between the conversational interface, analysis workflow, and document update workflow.

4.3 Used technologies

This section describes the technologies used in the implemented solution and explains why they were selected for the solution. The general explanation of the automation tools was discussed in Section 2.4.

Microsoft Copilot Studio was selected as the main component of the solution. It provides a conversational interface that allows the document maintainer to describe software changes to an agent with ease. Copilot supports human-in-the-loop interaction, which enables the maintainer to review and approve documentation changes before they are applied. This approval step was a requirement in the enterprise documentation workflow, since unwanted changes may introduce errors. Microsoft Copilot Studio integrates with other Microsoft products, making it suitable for an enterprise environment where these products are already widely adopted.

AI Builder is used to conduct LLM-based analysis within the solution. AI Builder offers prompt-based processing without the need for integrating external services. AI Builder is used to compare the user-provided content against existing documentation, identifying outdated or missing information, and creating update proposals.

User documentation is stored in Word format, and analysis or modifications cannot be performed directly without conversion. Document formatting is supported by the Encodian Word connector [22], which provides a collection of actions to modify and create Word documents. These actions provide the necessary functionality for converting Word documents into plain text and for controlled text replacement. This connector enables delegating document modification tasks to a service, allowing focus on the automation logic.

SharePoint [23] acts as the document repository for the solution. It functions as the source of documentation files and provides version control, which is crucial in enterprise documentation processes. SharePoint is also used to store backups of modifications generated by the solution, which enables rollbacks in case of er-

rors. SharePoint's integration with the Microsoft Power Platform allows seamless integration for both documentation analysis and update stages.

These technologies form the technology stack that supports AI-assisted documentation updates while remaining compatible with existing enterprise documentation processes.

4.4 Implementation stages

This section describes the practical implementation stages of the solution. The implementation was executed by developing the key components of the solution, which include Copilot Studio topic, analysis workflow, document update workflow, and AI Builder prompt.

These components were developed iteratively, since they depend on each other for the overall functionality of the solution. The Copilot Studio topic acts as the central component responsible for user interaction and managing the execution of workflows. The analysis workflow is responsible for retrieving the documentation and generating the update proposals, while the document update workflow applies the approved changes to the documentation. The AI Builder prompt is used to analyze the documentation content and generate update proposals based on the provided software changes.

After the components were developed, the solution was integrated into the organization's tools and tested using real documentation update scenarios.

4.4.1 Copilot Studio topic

The implemented solution begins with an entry point implemented using the Copilot Studio topic. The topic is responsible for coordinating user interaction, flow execution, and decision-making throughout the documentation update process. The topic

is responsible for collecting the input and iterating over proposed update changes.

The document update process begins when the user opens the conversational interface and provides software changes in natural language. The input usually consists of release notes or free-form text of the update. This input is stored as a variable and passed into the analysis workflow.

The analysis workflow processes the documentation and returns a structured output containing potential update proposals. This output contains information such as the relevant document filename, file path, original text segment, and the proposed replacement text. The returned data is parsed into a table, enabling iterative processing.

After the analysis workflow has processed the documents found inside the SharePoint document repository, the topic iterates through the table of proposed changes and presents them to the user. For each proposal, the document maintainer is prompted whether the proposed changes should be applied. If the user approves a change, the topic activates another flow, which is responsible for performing the document update. This flow applies the approved modifications to the relevant document and returns a message indicating whether the operation was successful, along with the details of the changes made. If the user declines the proposed change, the change is skipped, and the topic proceeds to the next proposal in the table. This ensures all the documentation changes remain under human control.

After all the proposed updates have been processed, the topic informs the user, and the document maintainer is given the option to start a new update cycle. If the user chooses to continue, the topic restarts the process by collecting new input; otherwise, the conversation ends.

This topic design allows a controlled update process, where artificial intelligence supports the analysis and proposes the updates, keeping the user in control of the modifications.

Figure 4.2 illustrates the high-level logic of the Copilot Studio topic. The topic coordinates the execution of analysis and update workflows while maintaining human control through iterative approval of update proposals.

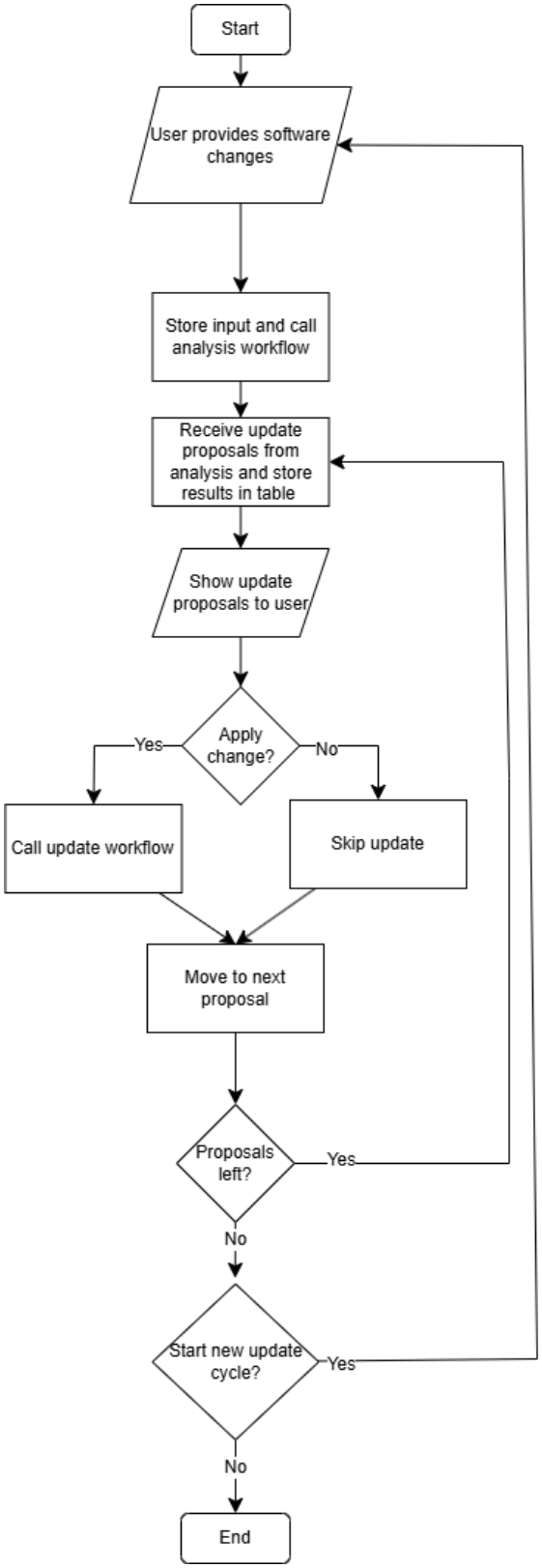

Figure 4.2: Flowchart of the Copilot Studio topic logic for iterative processing of documentation update proposals.

The implementation of the Copilot Studio topic required substantial effort. The topic functioned as the central mechanism of the solution, handling user interaction, coordinating multiple workflows and managing execution logic. One of the main challenges in implementing the topic was handling the structured outputs of the workflows. The proposed documentation changes were returned as structured data, which required manually defining the schemas and parsing logic, since Copilot Studio does not support complex data structures without manual configuration. To enable iterative processing of the modification proposals, the output had to be transformed into a table structure.

The Copilot Studio topic implementation highlighted that low-code tools simplify parts of the automation, but advanced usage still requires an understanding of the platform and tools. The final topic structure enabled a modular and extensible design of the documentation update process. The topic acts as an orchestration layer that separates the user interaction from processing logic, enabling the analysis and update workflows to be implemented and maintained as separate components.

4.4.2 Analysis workflow implementation

The analysis workflow is triggered when the topic mentioned earlier forwards the software change description provided by the user. Figure 4.3 illustrates the analysis workflow used to identify outdated documentation content and generate update proposals. The workflow begins by retrieving all the documentation files from the SharePoint document repository and assigning them to a variable for easier iteration. This allows analysis to always target the latest documentation versions.

After retrieving the documents, each document is processed in an iterative loop. For each document, the file content is retrieved from SharePoint. The retrieved document is stored in Word format, which needs to be converted into plain text for easier analysis. This conversion is performed using a custom HTTP-based document

processing solution that converts Word documents into a plain text version.

Once the plain text version of the document is available, the workflow passes the text into an AI Builder prompt to perform the analysis. The prompt is responsible for comparing the existing documentation with the software change description provided in the beginning. The goal of the prompt is to identify outdated content or missing information due to recent software changes.

The AI Builder prompt generates a structured output, which includes the relevant document filename, file path, whether replacement is needed, suggested replacement text, original text segment, and translation of replacement and original text. A conditional step is used to filter the prompt results, ensuring only the outputs that contain changes are appended into a variable for further processing.

After all the documents inside SharePoint have been analyzed, the analysis flow returns the collected update proposals to the Copilot Studio agent. The workflow completes without modifying files. The responsibility of deciding whether the proposed changes should be applied is delegated to the user.

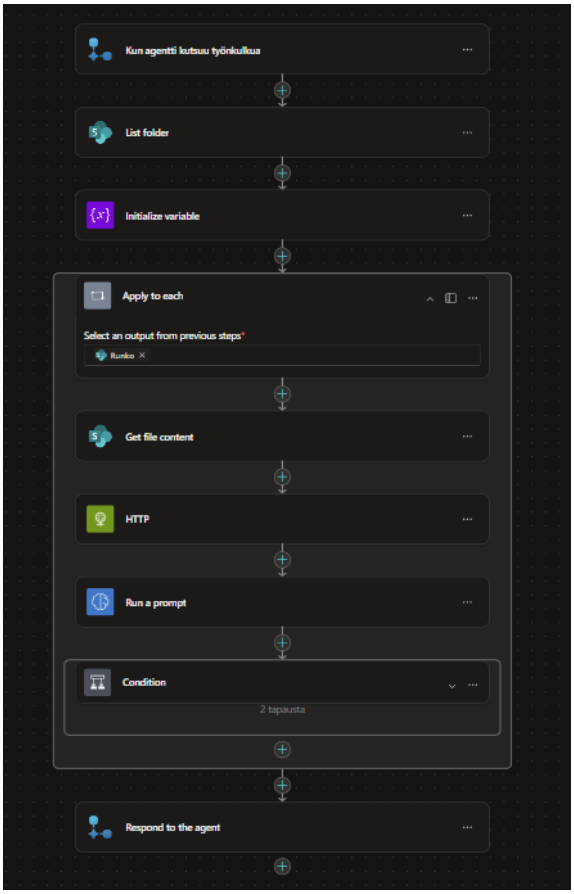

Figure 4.3: Analysis workflow for detecting outdated documentation content and generating update proposals based on software change description.

The analysis workflow is intentionally limited to identifying and proposing potential documentation updates. This modular design ensures a clear separation between workflows, allowing for easier problem fixing and scaling. This design choice preserves the human-in-the-loop and reduces the risk of unintentional changes that could be the result of fully automated text updates. This approach is essential when applying LLMs in enterprise documentation contexts.

4.4.3 Document update workflow implementation

The document update workflow is triggered by the topic if the user approves the proposed changes from the analysis workflow. The workflow starts by receiving a

structured data object that represents a single approved update. This object contains all the information in order to perform the modifications, such as the target filename, file path, original text segment, and the proposed replacement text. The workflow processes one update at a time, which enables more controlled and traceable document modifications.

Figure 4.4 illustrates the workflow, which begins by retrieving the target document from the SharePoint document repository. The retrieved file content is passed to an external document processing action from Encodian, which creates a separate copy of the document based on the provided search and replacement values. This action doesn't modify the file, but provides an output that can be used to overwrite the target document.

Before overwriting the original document, the workflow creates a backup of the file. The backup is stored in a separate folder in the same document repository using a timestamped filename, which ensures that previous versions can be restored if necessary. This step is crucial in reducing accidental modifications and supporting traceability, which is important in an enterprise documentation environment.

After the backup has been created, the target document is overwritten in SharePoint using the Encodian text replacement action output. A conditional check is then executed to verify whether the update operation was successful.

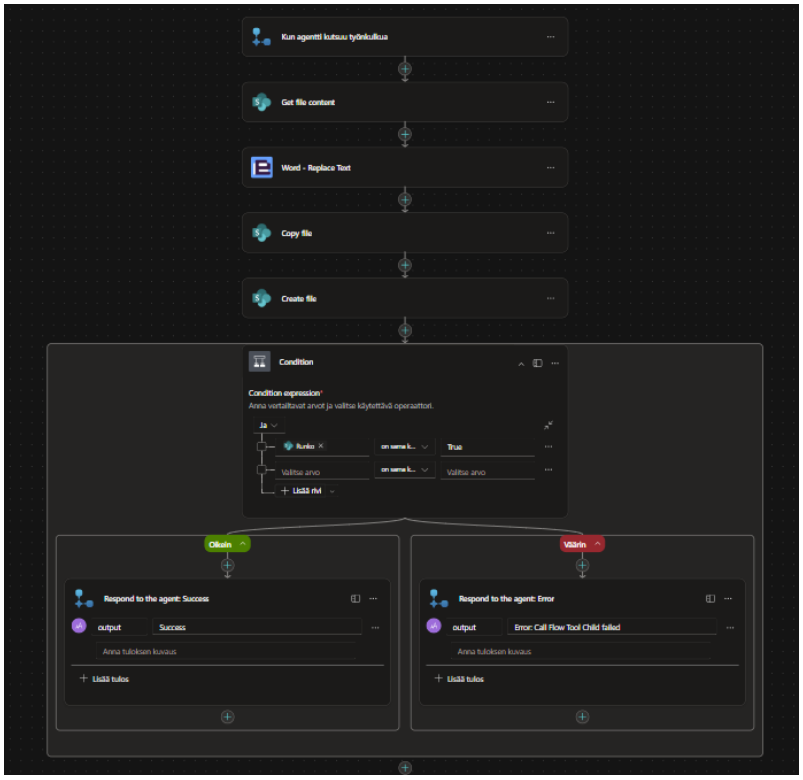

Figure 4.4: Documentation update workflow for applying approved document updates.

After the workflow completes, the topic informs the user of the modification outcome. The topic then proceeds to the next modification in the table, repeating the approval and update cycle until all the detected documents have been completed. Once all the modifications are done, the user is offered the option to start a new update cycle by providing a new software change description.

This design choice of iterative execution offers user control over each documentation modification. This choice aligns well with existing documentation modification practices, where each modification is reviewed and applied individually.

4.4.4 AI Builder prompt implementation

The AI Builder prompt is the core component of the analysis workflow, and it is responsible for identifying whether the given document requires changes, based on

the software changes given by the document maintainer.

Figure 4.5 presents the AI Builder prompt, which receives two inputs. The first input contains a description of the software changes that the user provides through the conversational interface. The second input contains the full content of a single documentation file after it has been converted from Word format into plain text using the Encodian action.

The task of the prompt is to locate a text segment that is related to the described software changes and would become outdated as a result of the software update. If such text exists, the prompt generates a structured JSON update proposal. This proposal includes the original text segment exactly as it appears in the document and a replacement text. The prompt forbids stylistic changes, paraphrasing, translation, or restructuring of surrounding text in order to minimize the unintended alterations.

To ensure a reliable output, the prompt is instructed to output a fixed JSON schema. This schema includes an UpdateNeeded field, which determines whether further processing should occur.

Additional validation rules are applied before the output. If the original text is not an exact substring of the document text, if the replacement text differs in language, or if the change is purely stylistic, the prompt is instructed to output a negative update result. Utilizing constraints and validation rules in prompts can reduce hallucinations and improve output reliability [24]. These additional validation rules reduce the risk of hallucinated update proposals and ensure only relevant changes are proposed to the user.

The prompt includes translation fields that are included only when the original documentation text differs from Finnish. This allows the user to have Finnish translations of the documentation text and suggested replacements.

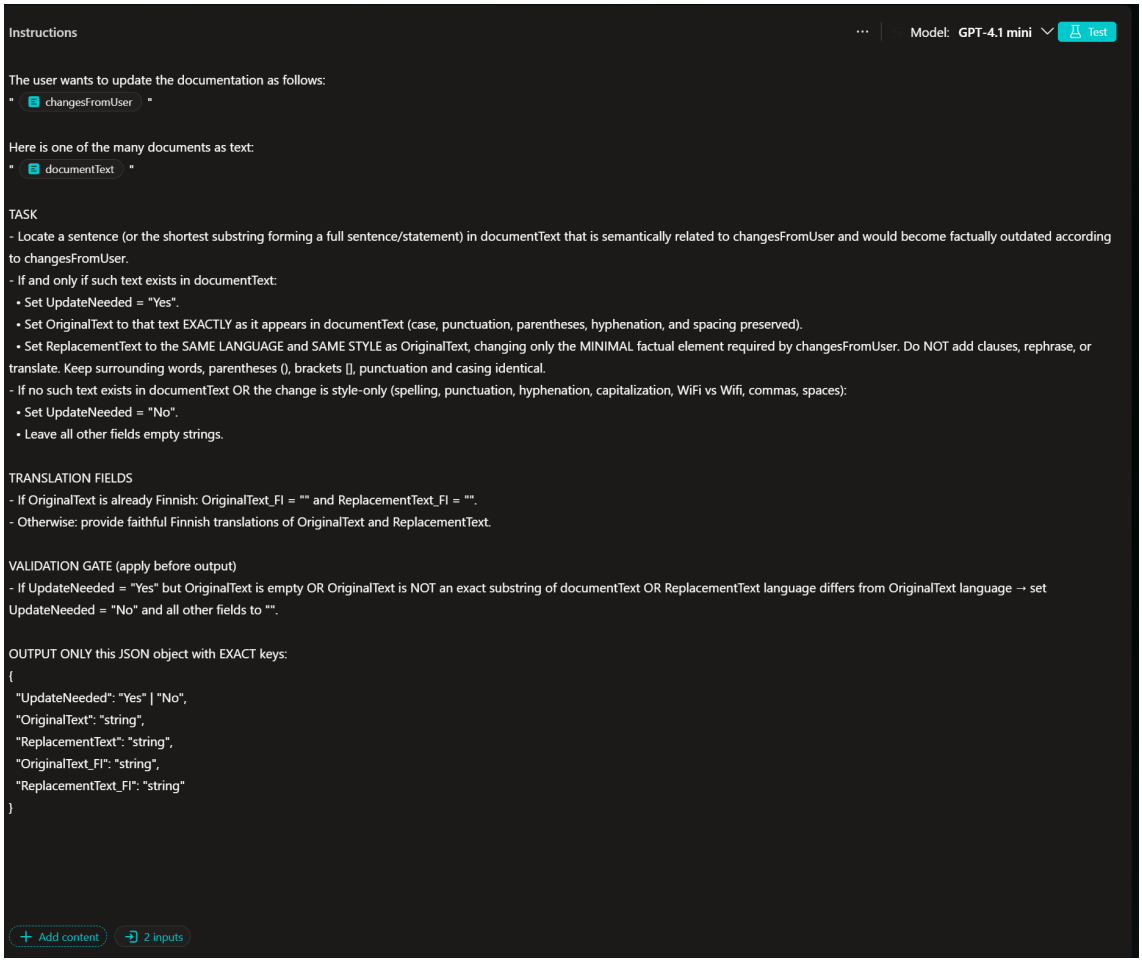

Figure 4.5: AI Builder prompt for analyzing documentation content and generating structured update proposals

Overall, the prompt is intentionally designed to be strict. The prompt's role is to assist the documentation maintainer by proposing minimal updates.

4.4.5 Integration and testing

Copilot Studio allows integration to multiple channels, such as websites and custom applications. The solution was integrated into Microsoft Teams to enable its use within the organization. Teams was selected due to its being a standardized platform within the organization that the documentation maintainer and the development

team widely use. This enabled access to the solution without introducing new tools or workflows.

Figure 4.6 illustrates the solution in use within Microsoft Teams. This shows an interaction where the user describes software changes, and the Copilot Studio agent presents the generated documentation update proposal. The proposal includes the affected document, original text segment, the suggested replacement segment, and their translation into Finnish. This allows the document maintainer to review the changes before accepting or rejecting them.

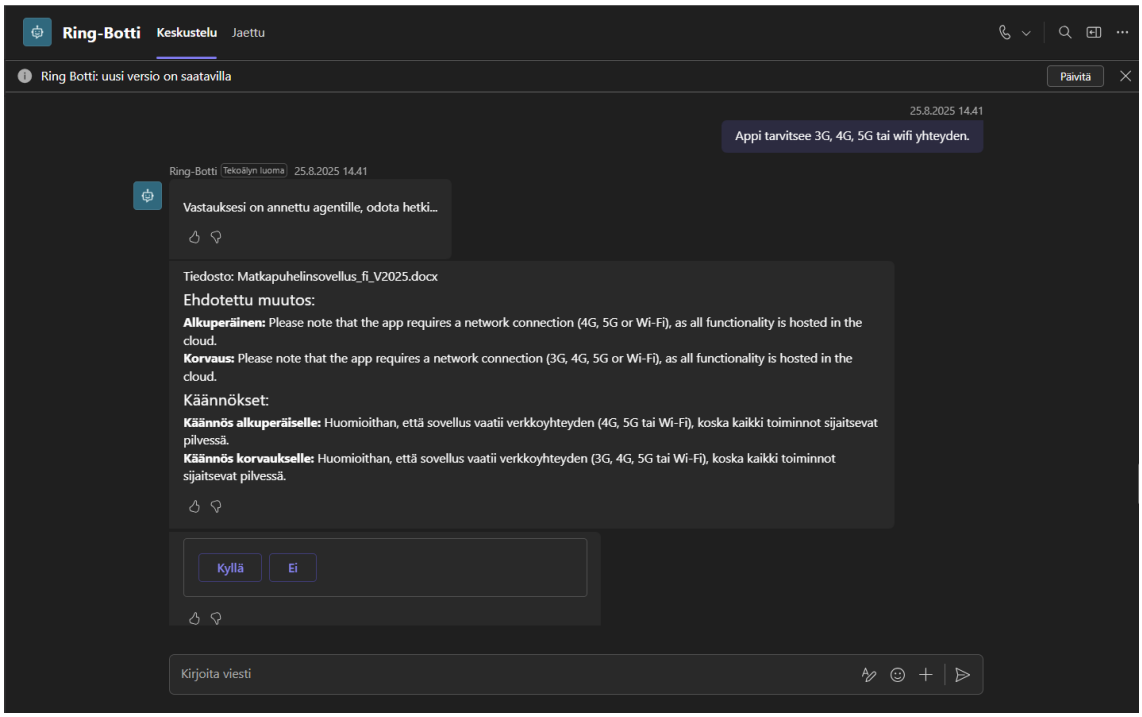

Figure 4.6: Solution integrated into Microsoft Teams, showing user interaction and generated update proposal.

Testing the solution was done during both development and integration phases. The solution was tested against real documentation files and realistic change descriptions provided from software release notes. The testing focused on ensuring that outdated documentation sections were correctly found, the proposed changes

were correct, and updates were applied accurately without affecting unrelated content.

After the solution was tested under normal conditions, error scenarios were tested. These scenarios included cases where no documentation updates were required, where the user rejected the proposed changes, and where document modification failed. These scenarios led to refinements in the Copilot Studio topic logic and workflow error handling. These refinements improved the robustness and user experience. Overall, the integration and testing phase concluded that the solution could be used to support document modification as a support tool within the enterprise documentation maintenance process.

5 Data collection and evaluation

This chapter presents the data collection and evaluation used in this thesis. The purpose of the evaluation is to examine how the implemented solution performs in a real documentation scenario used by the documentation maintainer.

The evaluation focuses on three metrics: the time required to perform the documentation updates manually and with the help of the AI solution, the accuracy of the AI-generated proposals, and whether manual editing was additionally needed.

5.1 Data collection

This section describes the data collected during the evaluation of the implemented solution. The goal of the data collection was to capture both quantitative and qualitative evidence of how the solution performs in real documentation update tasks.

The data for this case study were collected through real-world documentation update cases performed by the documentation maintainer within the organization. During the update process, the documentation maintainer recorded the effort required with both manual and AI-assisted updates.

For each case, the documentation maintainer performed the update task first manually and then with the implemented AI-assisted solution. This ensured that the AI-assisted tool did not influence the manual update process measurements. In both approaches, the time required to complete the update was recorded, and the

resulting documentation changes were recorded for comparison.

The primary dataset contains a set of documentation update cases based on real documentation update needs. This dataset is summarized in Table 5.1, which includes software changes such as requirement updates, feature additions, and small corrections to the document text.

Additionally, multiple metrics were collected to better evaluate the effectiveness and limitations of the solution. The purpose of the solution is not only to reduce documentation maintenance time, but also to support accurate and consistent updates. The evaluation contains four different aspects.

First, the manual documentation update time was recorded. This process included finding the relevant documentation file and text segment, modifying the text segment, and saving the updated version. The time required to complete this manual workflow is reported as "Manual time (min)" in Table 5.1.

Second, the same update was repeated using the AI-assisted solution. In this process, the maintainer provided the software change description to the Copilot Studio agent and waited for the solution to retrieve the documents, analyse them with the AI Builder prompt, and return the proposed updates. The reported time for this automated workflow is reported as "AI-assisted time (min)" in Table 5.1. This process included both the solution processing time and the human review before applying the changes. Additionally, if the solution proposal required manual revision, this was also included in the time measurement. The manual and AI-assisted time do not take into account the time to comprehend the changes made to the software based on the release notes. This approach enabled a fair comparison of both approaches.

Qualitative data were collected during the cases to evaluate the usefulness of the generated proposals. This was recorded in the "Correct suggestion" column in Table 5.1, indicating whether the proposal was correct, partially correct, or incorrect

Table 5.1: Documentation update times

Case	Update type	Manual time (min)	AI-assisted time (min)	Correct suggestion	Manual edits needed
1	Network requirement update	12	4	Yes	No
2	Web app feature functionality change	8	3	Yes	No
3	Swedish language document typo fix	3	3	Yes	No
4	Mobile app functionality change	9	5	Partly	Moderate
5	Mobile app troubleshooting section update	14	6	Yes	No
6	MFA Authentication update	12	7	Yes	Minor
7	User interface update	7	6	No	Yes
8	New functionality addition	18	-	No	Fully

by the maintainer. The column "Manual edits needed" in Table 5.1 describes the refinement effort required to fully align the documentation update with the software change and organizational documentation standards, ranging from no edits to minor or moderate adjustments.

Overall, the Table 5.1 provides evidence of time savings using the AI-assisted documentation update workflow.

5.2 Data Evaluation

This section evaluates the collected results and discusses how the AI-assisted tool performed in comparison to the traditional manual update process. The evaluation is structured in relation to the research questions by examining time efficiency, proposal correctness, and the overall maintainability impact of AI-assisted documentation updates.

Time efficiency

As shown earlier in Table 5.1 and illustrated in Figure 5.1, the AI-assisted approach yields a clear improvement in the time required to perform documentation updates. The largest time savings were observed when the maintainer needed to search longer documents to locate the relevant section.

For example, the software functionality changes typically required more manual effort due to their technical complexity. The AI-assisted workflow reduced the effort spent on finding the target document and drafting updates. Overall, the automa-

tion shifted the effort from finding documents and rewriting towards validation and decision-making.

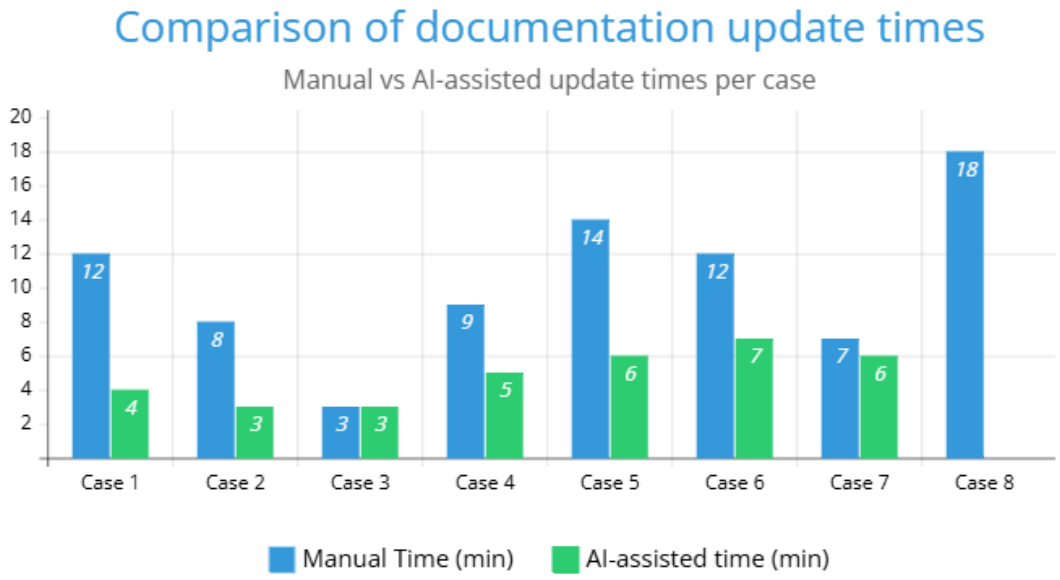

Figure 5.1: Comparison of manual and AI-assisted documentation update times per case

Overall, the results indicate that the AI-assisted approach improves time efficiency by reducing the effort required to locate the relevant documentation and generate updates.

Proposal correctness

In most cases, the documentation change proposals generated by the AI solution were correct. This indicates that the AI Builder prompt was, in most cases, capable of identifying the outdated content within the documents and generating meaningful changes. Certain cases were marked as partially correct or incorrect, meaning that the proposed changes required additional refinement by the maintainer. The incorrect cases often involved more complex changes, where the AI Builder prompt may

not have fully interpreted the software changes. Case 8, in Table 5.1, shows a failed attempt at adding new functionality to the documentation. This is marked with a dash because the solution was not able to identify that a new functionality had been added to the software and attempted to modify existing text segments within the documentation. This case indicates that the solution works best when modifying existing documentation rather than generating documentation for entirely new functionality. The partially correct cases involved documentation style or terminology that did not align fully with the requirements of enterprise documentation. This highlights the importance of treating LLM outputs as supportive suggestions rather than absolute truths.

Human-in-the-loop

The cases confirm that human approval remains a critical part of the workflow. Even when the proposals were correct, the organizational documentation quality standards required that the changes be reviewed and revised before being applied. The quality requirements for enterprise documentation are high because the documents are provided to customers. LLMs may produce incomplete or incorrect suggestions, especially when the software changes provided to the agent lack detail or when the documentation context is complex. Human approval acts as a safeguard against potential hallucinations or unintended modifications to the documentation, which could reduce documentation quality and reliability.

The evaluation of the cases previously included a case where the model did not produce a suitable update proposal and attempted to modify an unrelated section of the documentation. This case highlights why human approval is critical. The maintainer's review ensured that the incorrect suggestion was not applied to the documentation.

Overall impact of the solution on documentation maintainability

Based on the case results, the implemented AI-assisted solution demonstrates potential to improve the maintainability of the user documentation. The solution accelerates routine update tasks, which reduces the manual workload required from documentation maintainers.

Importantly, the solution does not replace the existing documentation practices but supplements them by functioning as a decision-support tool. The maintainer remains responsible for approving updates and ensuring that documentation quality remains aligned with organizational standards.

Overall, the evaluation shows that the solution can enhance the documentation process when integrated into the existing documentation update workflow. This is particularly highlighted in fast-evolving software projects, where manual documentation maintenance often struggles to keep up with software development changes.

User experience

Qualitative feedback was collected from the documentation maintainer during the case study. The maintainer reported that the solution was most useful in identifying relevant documentation files and text segments affected by the software changes. Locating the correct document from a large document repository requires significant manual effort. The ability to automatically analyze the software changes and find affected documentation reduced the update process effort.

Another benefit was the generation of update proposals, which provided a useful starting point for writing the final documentation update. Instead of writing the updated text from scratch, the maintainer could review and revise the suggested changes, which accelerated the documentation updates.

The documentation maintainer reported that the solution provided the best value when assisting in finding the relevant documentation and generating update sugges-

tions rather than performing automated modifications to the documentation. The maintainer preferred reviewing and revising the suggested update proposals before applying them to the documentation.

The usability of the solution was enhanced by its integration into Microsoft Teams. Teams is part of the everyday tools used by the documentation maintainer, which reduces the effort required to access the system and makes it easier to integrate into the existing documentation maintenance workflow.

6 Discussion

The previous chapters presented the empirical results of the implemented solution. The purpose of this chapter is to discuss the results of the case study and clarify the findings with reference to the research questions of this thesis. The objective of this thesis was to investigate how AI-assisted documentation maintenance can support user documentation in an enterprise environment.

The discussion reviews how the implemented solution affected the documentation update process compared to the traditional manual workflow. The comparison aims to evaluate the practical value of the solution, its limitations, and the role of human supervision in AI-assisted documentation maintenance.

6.1 Result analysis

The results in the previous chapter revealed that the AI-assisted workflow supports the documentation update process. The cases indicate that the implemented solution was able to reduce the overall effort required to perform documentation updates. On average, the AI-assisted approach reduced documentation update time by approximately 48% compared to the manual process. This aligns with previous research showing that AI-assisted documentation systems reduce documentation time by approximately 40% [14]. The calculation excluded the failed case, where the AI-assisted approach did not produce a valid update result, as no comparable time measurement could be obtained.

The most important observation from the evaluation is that the greatest time savings occurred during the document finding phase. In the manual documentation update process, the maintainer often spends most of their update time locating the relevant documentation file and section to be updated. The AI-assisted workflow was able to automate this step by analyzing the software change description and comparing it against the documentation content.

Another observation was that language models appear to perform particularly effectively in modifying existing documentation rather than generating entirely new content. In most of the cases, the generated proposals were correct or required minor modifications before approval. The evaluation included a case where the solution failed to recognize that the software change included a new functionality. Instead of proposing new documentation content, the solution focused on modifying text segments that were not relevant.

Additionally, the evaluation suggests that the quality of AI-generated update proposals was mainly comparable to the manually written updates in cases involving modifications to existing functionality. Most generated proposals only required minor refinements before approval, suggesting that the solution was able to preserve the terminology and structure of the original documentation. This suggests that AI-assisted documentation maintenance can help maintain documentation quality when human supervision is preserved.

The evaluation highlights the importance of maintaining human supervision in the workflow. Even though the solution was able to generate useful update proposals, the responsibility for documentation quality and accuracy remained with the documentation maintainer. Verification by the documentation maintainer ensures that the proposed updates follow organizational documentation standards and prevent incorrect information from being applied to the documentation. This approach is important in enterprise environments where documentation quality directly affects

the user experience and documentation reliability.

The performance of the AI-assisted solution can be related to the capabilities of LLM-based models discussed in Chapter 2.3. GPT models are designed to capture contextual relationships in text. This ability to capture context was reflected in the evaluation results, where the solution was able to identify relevant documentation text segments and generate update proposals. The results suggest that the contextual understanding capabilities of GPT-based models are effective when modifying existing documentation text segments. However, the case involving new functionality indicates that these models may struggle when the required changes extend beyond the existing context.

Overall, the results indicate that the AI-assisted solution supports the documentation maintainer by reducing manual effort and speeding up documentation updates. The results suggest that such systems currently function most effectively as decision-support tools rather than as fully autonomous systems for updating documentation.

6.2 Overall value of the solution

The implemented solution demonstrates practical value in supporting user documentation maintenance in enterprise environments. The evaluation results indicated improvements in documentation update efficiency, and the overall value of the solution is in its ability to support the documentation maintenance workflow.

Documentation maintenance requires the maintainer to understand software changes, identify affected documentation, and update it accordingly. In large enterprises, this process may require significant manual effort. Assisting the maintainer in identifying relevant documentation and generating update suggestions using the solution reduces the effort required during routine documentation maintenance tasks.

The productivity benefits of the solution are likely to become more apparent

when applied to a larger number of documentation update tasks. Although the evaluation cases in this study included a limited number of cases, the observed time savings may accumulate when documentation updates occur frequently.

Another aspect of the solution is that it supports the documentation maintainer rather than replacing them. The maintainer remains responsible for validating the suggested updates and ensuring that the documentation meets the organization's documentation standards.

The results suggest that AI-assisted documentation tools can improve documentation maintainability by reducing repetitive tasks and supporting the maintainer during the documentation update process.

6.3 Limitations of the solution

The implemented solution has several limitations that should be considered when analyzing the results of this study.

First, the evaluation included a limited number of documentation update cases. The dataset only included eight cases, which means that the results are indicative rather than generalizable. The documented time savings and performance improvements provide useful insight; further evaluation with a larger number of cases would be required to validate the results more extensively.

Second, the solution relies on the capabilities of LLMs, which may produce misleading or incorrect outputs. In some of the observed cases, the AI-generated proposals required manual corrections, and in one of the cases, the model failed to generate a suitable update. These observations highlighted the presence of hallucinations and the limitation of the model's ability to understand complex software changes. This limitation highlights the importance of human verification.

Another limitation is that the solution performs better when modifying existing documentation rather than generating entirely new documentation content. The

evaluation highlighted that the solution struggled in cases where new functionality was added.

The implemented solution relies on Microsoft Power Platform technologies, introducing a platform dependency; changes to services, such as pricing models or tool availability, may affect the solution's long-term usability and maintainability.

Another limitation is that effective use of the solution requires a documentation maintainer with sufficient domain knowledge. Since the AI-generated suggestions may not always be fully accurate or meet the documentation standards, the maintainer must be able to evaluate and refine the proposed updates. This limits the broader applicability of the solution.

Finally, the user experience was based on feedback from a single documentation maintainer. The feedback provided was valuable for the practical usability of the solution; a larger user evaluation would be required to make more general conclusions about usability and user experience.

Overall, these limitations suggest that further development and evaluation are required before the solution can be applied more broadly within the organization.

6.4 Answers to research questions

The purpose of this section is to provide explicit answers to the research questions of this thesis based on the results of the case study and the analysis presented in the previous sections. The answers are based on observations made during the implementation and use of the solution and on the empirical evaluation.

RQ 1: How does artificial intelligence-assisted automation change the process of updating user documentation compared to manual methods?

The results of this study show that AI-assisted automation changes the documentation update process by shifting the focus from manual searching and writing towards

validation and decision-making. This conclusion is based on the empirical results of the case study, where the most significant time savings were observed in the document finding phase.

In the traditional manual workflow, the documentation maintainer spends a significant portion of their time locating all the affected documentation and writing the updated documentation. The AI-assisted approach reduces this effort by finding relevant documentation sections and generating update proposals. This finding is in line with previous research, which has shown that AI-assisted documentation systems can significantly reduce documentation time [14].

RQ 2: How can language models be used effectively to support the identification and implementation of user documentation?

The results of this study show that language models can be used effectively to analyze software change descriptions and compare them with existing documentation content to identify documentation sections affected by software changes. This conclusion is based on the empirical observations made during the implementation and evaluation of the solution.

Language models can generate draft update proposals that assist the documentation maintainer in implementing the required documentation changes. The results also indicate that language models are most effective when applied to modifying existing documentation rather than generating entirely new content.

The results also highlight the importance of human supervision. Language models can automate parts of the process, but the final validation of the generated documentation should remain the responsibility of the documentation maintainer to ensure the documentation stays aligned with organizational standards and factually correct. This finding aligns with recent research on AI-in-the-loop systems, where humans make the final decision while AI systems provide supportive suggestions and

assistance [25].

RQ 3: What are the advantages and limitations artificial intelligence and automation bring to the maintainability of user documentation?

The main advantages of AI-assisted documentation maintenance include improved efficiency, reduced manual effort, and faster documentation updates. By automating the most time-consuming parts, such as identifying relevant documentation and generating update suggestions, the solution supports the documentation maintainer and improves the overall documentation maintenance workflow.

However, the results also highlight several limitations that were discussed in earlier chapters. Language models may produce incorrect or misleading suggestions, which require human verification. This limitation is in line with existing research on language model hallucinations and reliability [11]. The solution performs better when modifying existing documentation than generating new content. Even though the AI-assisted solution can improve documentation maintainability, it should be used in a way that supports the workflow and not by replacing it.

6.5 Future improvements

Based on the results of this study, several improvements can be identified for the implemented solution.

The primary direction for future development is to shift the solution's focus from modifying documentation to supporting the documentation maintainer. The results indicate that the solution provides the most value when assisting the maintainer in locating relevant documentation and text sections, and generating update suggestions. Instead of directly modifying the documentation, it could focus on identifying relevant sections and providing update suggestions as a starting point.

Another future improvement is refining the language model prompts used in

the solution. Improved prompt design may help the solution better understand software changes and generate more accurate update suggestions. The solution was not able to distinguish between modifications to existing functionality and entirely new functionality, which may be due to the existing prompt design.

Further improvements could be achieved by conducting additional evaluations with a larger number of documentation update cases. A larger dataset would provide more reliable insights into the usefulness of the solution and allow a more thorough assessment of its performance in different documentation update scenarios.

Another important direction for future work is to evaluate the impact of different language models on the performance of the solution. The evaluation indicated that the current model struggled to identify when new functionality was introduced and attempted to modify existing documentation.

Future research could involve a comparative evaluation of different language models to determine whether the latest models are better at recognizing between modifications and new functionality. This could improve the solution's ability to generate more accurate update proposals, particularly in complex update scenarios.

7 Conclusion

This thesis investigated how an artificial intelligence-assisted workflow can support the process of maintaining user documentation in enterprise environments. The study focused on evaluating the effectiveness of large language models in identifying outdated documentation and assisting the documentation maintainer in generating update proposals based on software change descriptions.

The results of the study show that an AI-assisted solution improves the efficiency of documentation maintenance, reducing update time by approximately 48%. The most significant improvement was detected in the document search phase, where the solution reduced the effort required to locate relevant documentation and text segments. The language models were able to generate useful update proposals in most cases, particularly when modifying existing content. The findings also highlight that AI-based solutions for enterprise documentation maintenance are not fully reliable when used as fully autonomous systems. The evaluation indicated that the models may produce incomplete or incorrect suggestions, especially in cases involving complex changes or new functionality. As a result, human supervision remains a critical component of the documentation maintenance process.

Overall, the study demonstrated that AI-assisted tools are most effective when used as a support tool rather than a fully automated solution. By combining the strengths of artificial intelligence with human expertise, organizations can improve documentation maintainability while ensuring quality and reliability.

AI-assisted automation introduces a promising approach to supporting documentation maintenance in modern software development environments. As AI capabilities continue to advance, their role in enterprise documentation processes is likely to expand, enabling more efficient and scalable maintenance workflows.

References

[1] H. Birru, A. Cicchetti, and M. Latifaj, "Supporting automated documentation updates in continuous software development with large language models", Jan. 1, 2025, pp. 92–106. DOI: 10.5220/0013286800003928.

[2] M. Gastegger and S. Zünd, "Maintenance of technical and user documentation", Apr. 17, 2015.

[3] "The state of AI: Global survey 2025 | McKinsey". [Online]. Available: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai.

[4] W. X. Zhao et al., *A survey of large language models*, Jun. 12, 2023. DOI: 10.48550/arXiv.2303.18223. arXiv: 2303.18223[cs]. [Online]. Available: http://arxiv.org/abs/2303.18223.

[5] "AI-powered low-code tools | microsoft power platform", Accessed: May 12, 2026. [Online]. Available: https://www.microsoft.com/en-us/power-platform.

[6] H. Naveed et al., *A comprehensive overview of large language models*, Oct. 17, 2024. DOI: 10.48550/arXiv.2307.06435. arXiv: 2307.06435[cs]. [Online]. Available: http://arxiv.org/abs/2307.06435.

[7] A. Vaswani et al., *Attention is all you need*, 2017. DOI: 10.48550/arXiv.1706.03762. arXiv: 1706.03762[cs]. [Online]. Available: http://arxiv.org/abs/1706.03762.

[8] S. T. Piantadosi and F. Hill, "Meaning without reference in large language models", no. arXiv:2208.02957, Aug. 12, 2022. DOI: 10.48550/arXiv.2208.02957. arXiv: 2208.02957[cs]. [Online]. Available: http://arxiv.org/abs/2208.02957.

[9] J. Wu et al., *Can large language models understand uncommon meanings of common words?*, May 9, 2024. DOI: 10.48550/arXiv.2405.05741. arXiv: 2405.05741[cs]. [Online]. Available: http://arxiv.org/abs/2405.05741.

[10] S. Minaee et al., *Large language models: A survey*, Mar. 23, 2025. DOI: 10.48550/arXiv.2402.06196. arXiv: 2402.06196[cs]. [Online]. Available: http://arxiv.org/abs/2402.06196.

[11] L. Weidinger et al., "Taxonomy of risks posed by language models", in *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '22, New York, NY, USA: Association for Computing Machinery, Jun. 20, 2022, pp. 214–229, ISBN: 978-1-4503-9352-2. DOI: 10.1145/3531146.3533088. [Online]. Available: https://dl.acm.org/doi/10.1145/3531146.3533088.

[12] J. Y. Khan and G. Uddin, *Automatic code documentation generation using GPT-3*, Sep. 6, 2022. DOI: 10.48550/arXiv.2209.02235. arXiv: 2209.02235[cs]. [Online]. Available: http://arxiv.org/abs/2209.02235.

[13] C. Luca, "Natural language processing (NLP) for document analysis", Apr. 1, 2025.

[14] H. Ju et al., "Generative AI-based nursing diagnosis and documentation recommendation using virtual patient electronic nursing record data", *Healthcare*

Informatics Research, vol. 31, no. 2, pp. 156–165, Apr. 2025, ISSN: 2093-3681. DOI: 10.4258/hir.2025.31.2.156.

[15] J. L. Gastaldi, J. Terilla, L. Malagutti, B. DuSell, T. Vieira, and R. Cotterell, *The foundations of tokenization: Statistical and computational concerns*, 2025. arXiv: 2407.11606 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2407.11606.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding", no. arXiv:1810.04805, May 24, 2019. DOI: 10.48550/arXiv.1810.04805. arXiv: 1810.04805[cs]. [Online]. Available: http://arxiv.org/abs/1810.04805.

[17] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training", 2018. [Online]. Available: https://www.semanticscholar.org/paper/Improving-Language-Understanding-by-Generative-Radford-Narasimhan/cd18800a0fe0b668a1cc19f2ec95b5003d0a5035.

[18] S. Mandvikar, "Augmenting intelligent document processing (IDP) workflows with contemporary large language models (LLMs)", *International Journal of Computer Trends and Technology*, vol. 71, pp. 80–91, Oct. 30, 2023. DOI: 10.14445/22312803/IJCTT-V71I10P110.

[19] G. A. Cutting and A.-F. Cutting-Decelle, *Intelligent document processing – methods and tools in the real world*, Dec. 28, 2021. DOI: 10.48550/arXiv.2112.14070. arXiv: 2112.14070[cs]. [Online]. Available: http://arxiv.org/abs/2112.14070.

[20] R. Kolandaisamy, H. Rajagopal, I. Kolandaisamy, and G. Sinnappan, "The smart document processing with artificial intelligence", *Proceedings of International Conference on Artificial Life and Robotics*, vol. 29, pp. 534–540, Feb. 22, 2024. DOI: 10.5954/ICAROB.2024.OS18-8.

[21] Microsoft. "Power your ai transformation with microsoft copilot studio". [Online]. Available: https://marketingassets.microsoft.com/gdc/gdcJZPIov/original.

[22] Microsoft. "Encodian - word". [Online]. Available: https://learn.microsoft.com/en-us/connectors/encodianword/.

[23] "Collaborative content management, and secure file sharing | microsoft SharePoint", Accessed: May 12, 2026. [Online]. Available: https://www.microsoft.com/en-us/microsoft-365/sharepoint/collaboration.

[24] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, *A systematic survey of prompt engineering in large language models: Techniques and applications*, Mar. 16, 2025. DOI: 10.48550/arXiv.2402.07927. arXiv: 2402.07927[cs]. [Online]. Available: http://arxiv.org/abs/2402.07927.

[25] S. Natarajan, S. Mathur, S. Sidheekh, W. Stammer, and K. Kersting, "Human-in-the-loop or AI-in-the-loop? automate or collaborate?", no. arXiv:2412.14232, Dec. 18, 2024. DOI: 10.48550/arXiv.2412.14232. arXiv: 2412.14232[cs]. [Online]. Available: http://arxiv.org/abs/2412.14232.