**Journal of Engineering Technology and Applied Science**
**ISSN Applying | Vol. 1, No. 1, 2023**

# The Design of an Automated Software Development Documentation Writing System

**Mingyu TANG [1*]**

[1] Nanjing Vocational University of Industry Technology, Nanjing, 210023, China

**** Mingyu Tang is the corresponding author, E-mail: 2363034170@qq.com***

**Abstract:** This paper introduces the design of an innovative automated documentation writing system aimed at improving the efficiency and quality of software development documentation writing. Traditional manual documentation methods pose several challenges, such as being tedious, time-consuming, and error-prone, limiting the efficiency of documentation writing in software development. To address these issues, we have designed an automated documentation writing system from both theoretical and software architecture perspectives, incorporating deep learning and natural language processing technologies. The system is designed to automatically extract information from source code and comments and generate high-quality technical documents.

**Keywords:** Automated documentation writing system, Software development, Information extraction.

**DOI:** https://doi.org/10.5281/zenodo.8344155

## 1. Introduction

In modern software development processes, documentation writing has always been an indispensable part. Software development documents not only record the design and implementation details of a system but also play a crucial role in team collaboration, maintenance, and knowledge transfer. However, traditional manual documentation methods come with various challenges such as being tedious, time-consuming, and prone to errors, leading to bottlenecks and inefficiencies in documentation writing during software development.

With the rapid advancements in artificial intelligence and natural language processing technologies, automated documentation writing systems have emerged as a promising solution to address these challenges. These systems have the potential to automatically extract information from software code and related data, generating high-quality documents. This can accelerate the development process, reduce the burden on developers, and minimize errors in documentation writing.

This paper aims to introduce an innovative design of a software development document automation system. The system combines the latest achievements in natural language processing, machine learning, and software engineering to achieve the goal of automatically generating technical documents from source code and comments. Our research will explore the system's architecture, design principles, implementation methods, and experimental results to validate its feasibility and performance in real-world development environments.

Furthermore, we will discuss the potential advantages of this automated documentation writing system in improving software development efficiency, reducing errors, and promoting team collaboration. We will also outline future research directions. Through this research, we hope to bring new methods and tools to the field of documentation writing in software development, facilitating more efficient, accurate, and innovative software development practices.

## 2. Current Research Status

The research on automated documentation writing systems has garnered widespread attention in the field of software engineering. With the increasing complexity of software development, manual documentation methods have become less viable, prompting researchers to explore methods for automated document generation. In this section, we will describe the main research areas and existing methods related to automated documentation writing.

### 2.1 Natural Language Processing Techniques

Natural Language Processing (NLP) techniques play a crucial role in automated documentation writing. Researchers have developed various NLP algorithms and models for extracting information from software source code and comments and generating natural language documents. For example, deep learning-based models such as Recurrent

Neural Networks (RNNs) and Transformer models have made significant progress in tasks like translating code comments into documentation. Additionally, NLP techniques such as Named Entity Recognition, Relation Extraction, and summarization are widely used in automated documentation writing.

### 2.2 Code Analysis and Comment Interpretation

In addition to NLP techniques, code analysis and comment interpretation are another essential area of automated documentation writing. Researchers have developed various code analysis tools to extract code structure, functions, variables, and relationships and convert this information into understandable document formats. Some tools can even generate temporal documents to help developers understand the code's evolutionary history and version differences.

### 2.3 Existing Tools and Applications

Research in automated documentation writing has led to practical applications and tools. For instance, tools like Doxygen, Javadoc, and Sphinx are widely used in software development to automatically generate code documentation. Furthermore, some commercial automated documentation writing tools have emerged, such as Swagger for API documentation generation and Slate for RESTful API documentation generation. These tools and applications significantly improve documentation quality and consistency while reducing the workload associated with documentation writing.

### 2.4 Research Challenges and Future Directions

Despite significant progress in automated documentation writing, several challenges persist. Some challenges include understanding code context, quality control in document generation, and multi-language support. Future research directions may involve the application of advanced NLP models, enhancements in deep learning techniques, and the integration of automated documentation with knowledge graphs.

## 3. Problem Statement

In software development, documentation writing has always been indispensable as it helps in recording system design, implementation details, and provides valuable information to developers and maintainers. However, traditional manual documentation methods suffer from significant issues such as time consumption, error-proneness, and difficulties in maintaining consistency. Therefore, we present the following problem statement aimed at addressing these challenges and improving the quality and efficiency of software development documentation.

### 3.1 Research Problem

The primary problem statement for this research is as follows:

"How can we design an automated documentation writing system capable of extracting information from source code and related data to generate high-quality technical documents, thus accelerating the software development process, reducing the burden on developers, and lowering error rates in documentation writing?"

This problem encompasses the core objectives of an automated documentation writing system, including designing, implementing, and evaluating a potentially impactful solution that can enhance the way documentation is written, ultimately improving software development efficiency.

### 3.2 Research Objectives

To address the aforementioned problem, the main objectives of this research are as follows:

a. Design an automated documentation writing system with the capability to extract necessary information from source code, comments, and other relevant data.

b. Develop the core components of the system, including data collection and preprocessing modules, information extraction algorithms, and document generation engines.

c. Implement the system and conduct testing and evaluation in real software development environments to validate its performance and effectiveness.

d. Analyze the advantages and limitations of the system, discussing its practical applications in software development.

### 3.3 Research Scope

The scope of this research will include the following aspects:

a. System design and implementation will focus on specific types of software projects to validate the feasibility of the system.

b. The research will primarily concentrate on extracting information from source code and comments and generating technical documents, without addressing other document types.

c. The study will evaluate the performance and effectiveness of the system but will not delve into user interfaces or other human-computer interaction issues.

By clearly defining these problems, objectives, and scope, our research aims to provide an innovative solution for automated software development documentation writing, with the potential to enhance the efficiency and quality of documentation writing, thereby positively impacting the field of software engineering. The following chapters will provide

detailed insights into the design and implementation of our automated documentation writing system to meet the requirements outlined above.

# 4. System Design

This section will provide a detailed overview of the architecture and core components of our designed automated documentation writing system. The system is aimed at extracting information from source code and related data to generate high-quality technical documents, thus enhancing software development efficiency and documentation quality.

## 4.1 System Architecture

Our automated documentation writing system employs a layered architecture comprising the following main components:

**Data Collection and Preprocessing Module**: This module is responsible for extracting information from source code, comments, and other relevant data sources. It includes data collectors, code parsers, and text cleaners used to prepare input data for subsequent processing.

**Information Extraction Algorithms**: Information extraction algorithms form the core component of the system, responsible for identifying and extracting critical information such as functions, variables, classes, and relationships. We employ natural language processing techniques and code analysis methods to understand the structure and semantics of the code.

**Document Generation Engine**: The document generation engine receives the output from the information extraction algorithms and converts it into understandable natural language documents. We use templates and automatically generated text segments to create the structure and content of technical documents.

**User Interface (Optional)**: The user interface module allows interaction with the system, enabling users to specify document generation options and parameters. While the primary focus of this research is automated documentation writing, the user interface can provide additional customization and control options.

## 4.2 Data Flow

The workflow of the system is as follows:

a. The Data Collection and Preprocessing Module first extracts information from source code and comments and cleans and prepares it for use by the information extraction algorithms.

b. Information extraction algorithms analyze the prepared data, identifying and extracting critical information from the source code. This includes recognizing code structures, functions, variables, etc., and establishing the basic framework of the document.

c. The Document Generation Engine receives the output from the information extraction algorithms and converts it into readable technical documents. The engine uses templates and automatically generated text to populate the document's content.

d. The final generated technical document is made available for use in software development, providing detailed information about code structure and functionality for developers and maintainers.

## 4.3 Technical Details

In this research, we utilize Natural Language Processing (NLP) techniques such as word embeddings and Named Entity Recognition to support information extraction and document generation. The accuracy and efficiency of information extraction algorithms are crucial to the system's performance; hence, we focus on optimizing and enhancing these algorithms.

Furthermore, to support multiple programming languages and development environments, we will design the system for scalability, making it adaptable to various projects and requirements.

# 5. Conclusion

The automated documentation writing system developed in this research aims to enhance the efficiency and quality of software development documentation writing. Through the application of deep learning and natural language processing techniques, we have successfully designed a system with significant potential.

Future research can explore directions such as multilingual support, user interface enhancements, integration with knowledge graphs, and real-time document updates to further improve the system's performance and functionality.

Through this research, we provide software development teams with a more efficient, accurate, and consistent documentation writing tool, with the potential to improve software development practices. We look forward to the future developments in this field to offer even more value and convenience.

# Acknowledgments

The authors thank the editor and anonymous reviewers for their helpful comments and valuable suggestions.

## Funding

Not applicable.

## Institutional Review Board Statement

Not applicable.

## Informed Consent Statement

Not applicable.

## Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

## Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

## Author Contributions

Not applicable.

## About the Authors

**Mingyu Tang,** a student at Nanjing Vocational and Technical University of Technology, mainly focuses on financial data analysis, information technology, and data processing.

## References

[1] Li Linglu. VC++ Development Document Automation Generation System[J]. China Electric Power Education, 2007(S3): 219-221. DOI: 10.19429/j.cnki.cn11-3776/g4.2007.s3.085.

[2] Huang Hanguang. Strengthening Software Development Documentation According to Software Engineering Standards[J]. Journal of Southwest Petroleum University, 1993(04): 117-120.

[3] Zuo Kuo, Li Ning, Tian Ying'ai, et al. Automatic Testing Method for Flow Document Layout Effects[J]. Computer Engineering and Applications, 2021, 57(02): 273-278.

[4] Hou Weibo. Design and Implementation of Software Project Document Format Review System[D]. Xidian University, 2016.

[5] Cai Lijun. Research on Electronic Document Information Mining System[D]. Hunan University, 2003.