ISSN 2093-0542




KOREAN SOCIETY FOR INTERNET INFORMATION

The 20th Asia Pacific International Conference on Information Science and Technology

(APIC-IST 2025)

July 06-09, 2025, SAii Laguna Resorts, Phuket, Thailand

http://www.apicist.org

Proceedings of APIC-IST 2025

| Organized by |

Korean Society for Internet Information (KSII)

Contents

Automatic Requirements Registration Mechanism

Yejin Jin and R. Young Chul Kim*
SE Laboratory, Hongik University
Seoul, South Korea
[e-mail: yejin_jin@g.hongik.ac.kr, bob@hongik.ac.kr]
*Corresponding author: R. Young Chul Kim

Abstract

The Software Requirements Specification (SRS) is an essential document in the early phases of software development. However, the existing methods of manually analyzing documents and registering tasks in the Issue Tracking System consume a lot of time and human resources. To solve this, we propose an automated mechanism to classify functional requirements from the SRS and integrate it with an Issue Tracking System, such as Redmine. For this, we 1) extract functional requirements by learning the classification model using the PURE (Public Requirements) dataset, 2) register the result as an Issue in Redmine, 3) automatically generate UML design from natural language based on the registered Issues, and 4) register the generated design in Redmine's Wiki and link it with existing Issues. With this, we can guarantee the maintenance of requirement traceability among requirements, designs, and code. Finally, we may increase development productivity.

***Keywords*:** Software Development Process, Functional Requirements, Issue Tracking System

1. Introduction

The Software Requirements Specification (SRS) is written through communication between stakeholders, based on customer needs [1]. SRS includes system overview, user requirements, functional requirements, and non-functional requirements. Software development is designed, developed, and tested with a focus on the functional requirements outlined in the SRS. These functional requirements are used for issue management and implementation planning in the software development cycle. Therefore, it is essential to classify and understand them accurately. Recently, research has been conducted to classify non-functional requirements (NFR) and functional requirements (FR) from SRS using machine learning and natural language processing technologies [2, 3]. However, most research focuses on improving the accuracy of classification models. There is a lack of research on applying analyzed requirements to the software development process. Many software developers utilize the Issue Tracking System to convert functional requirements into actual development tasks [4]. The Issue Tracking System enables systematic software development and collaboration, task allocation, and task progress tracking. However, to utilize this, developers must directly interpret the SRS document and manually create issues based on it. This manual work requires a significant amount of time and costs [5].

Therefore, we propose a mechanism that

This research was conducted with the support of the Korea Creative Content Agency (Project Name: Artificial Intelligence-Based Interactive Multimodal Interactive Storytelling 3D Scene Authoring Technology Development, Project Number: RS-2023-00227917, Contribution Rate: 100%) and the Korea Research Foundation's four, Brain Korea 21 (Project Name: Ultra-Distributed Autonomous Computing Service Technology Research Team, Project Number: 202003520005).

classifies functional requirements from the SRS and links them to the Issue Tracking System to automatically register and track issues. In addition, we conducted a study on automatically generating designs by consistently analyzing natural language requirements in a previous study [6]. We apply this in this study to connect SRS to the design model. This approach can contribute to improved consistency between requirements and designs, as well as enhanced development productivity through automation. In addition, when requirements change, they can be automatically tracked and reflected in the design. Therefore, the efficiency of change management is expected to improve.

Section 2 describes related research, including prior research on requirements classification through AI learning, and explains the Issue Tracking System used in this study. Section 3 describes the mechanism of this study. Finally, Section 4 mentions our conclusion.

2. Related Works

2.1 Approaches to Software Requirement Classification Using Artificial Intelligence Learning

Accurate requirements analysis is essential for developing high-quality software. Research on automatically classifying software requirements has been ongoing, and recently, deep learning and pre-trained models have been utilized.

Khayashi applied various deep learning algorithms to classify functional and non-functional requirements [2]. We extracted and labeled requirements using the PURE (Public Requirements) data set. We then extracted features using various deep learning models (LSTM, BiLSTM, GRU, and CNN) and embeddings (Keras and GloVe). Additionally, ensemble learning based on Hard/Soft Voting was applied to enhance the performance of requirements classification.

In another study, Rahman proposes a feature extraction algorithm based on pre-trained embedding models (GloVe, Word2Vec, FastText) to classify non-functional requirements in limited data environments [3]. This study classifies non-functional requirements into five categories by utilizing the International RE Conference 2017 Challenge and PROMISE datasets.

Studies that apply learning models to requirements classification only consider the accuracy of the model. However, there are very few cases in which this classification has been used in actual work. Therefore, we apply it to the Issue Tracking System to improve developers' work efficiency and reduce costs.

2.2 Issue Tracking System

The software development life cycle consists of the following phases: Requirements Analysis, Design, Development, and Testing. Each phase produces an output, and each output should be stored with version control to ensure traceability [7]. An issue-tracking system is commonly used for this purpose. Most developers manage the entire development process through an issue tracking system. The Issue Tracking System registers various bugs, tasks, and feature requests that occur in software development or project management. This tool can enhance the quality of software products by improving collaboration efficiency and enabling tracking of bugs and feature change requests. Additionally, since it is integrated with project management, it provides a comprehensive understanding of the overall project flow, encompassing schedule management and resource allocation.

Table 1. Comparison of Issue Tracking Systems

Category	Jira [8]	GitHub Issues [9]	Redmine [10]
Type	Commercial ITS	It's built into GitHub	Open Source ITS
CI / CD Integration	Bitbucket, GitHub, GitLab	GitHub Actions	Jenkins, Hook, …
Project Management	Advanced	Basic	Moderate
Customize	High	Low	Very High
Learning Curve	High	Low	Moderate

Issue Tracking Systems include tools such as Jira, Redmine, and GitHub Issues. These tools support various functions, including task

registration, status tracking (such as new, in progress, or completed), team member assignment, deadline setting, and issue filtering. **Table 1** shows the features of each tool.

Jira has a wide range of features and is suitable for large companies. However, very complex control is required. GitHub Issues is optimized for Git-centric collaboration but provides limited customization options. Redmine, although it features a relatively outdated user interface compared to other tools, is free, open-source, and supports various plugins, offering powerful customization options. Therefore, Redmine is selected for this study due to its flexibility in adapting to the proposed research workflow.

3. Automatic Task Assignment Mechanism from Software Requirements Specification

We develop a software development process based on SRS documents. The proposed method extracts functional requirements through AI learning and assigns them to issues in the Issue Tracking System. Once each function is saved as an issue, a design drawing is created based on it.

Our mechanism goes through four procedures. 1) Extract functional requirements from SRS documents written in natural language using AI, 2) Register the extracted functional requirements as issues in Redmine, 3) Generate a UML design based on the registered issues according to the process of previous studies, and 4) Register the generated design image in Redmine Wiki. Even if the SRS document is revised, the basic process procedure remains unchanged. In this case, to maintain traceability of requirements and designs, the existing UML design is linked to the existing issue, and the new UML design is related to the latest issue.

3.1 Functional Requirements Classification

We extract functional requirements from SRS documents using AI. Documents can be divided into functional requirements, non-functional requirements, and introduction, and supervised learning-based text classification is possible. Sentences are extracted from SRS documents, and unnecessary symbols are removed through a preprocessing process. Each sentence is vectorized through text embedding, and the generated sentence vector is input to a trained AI

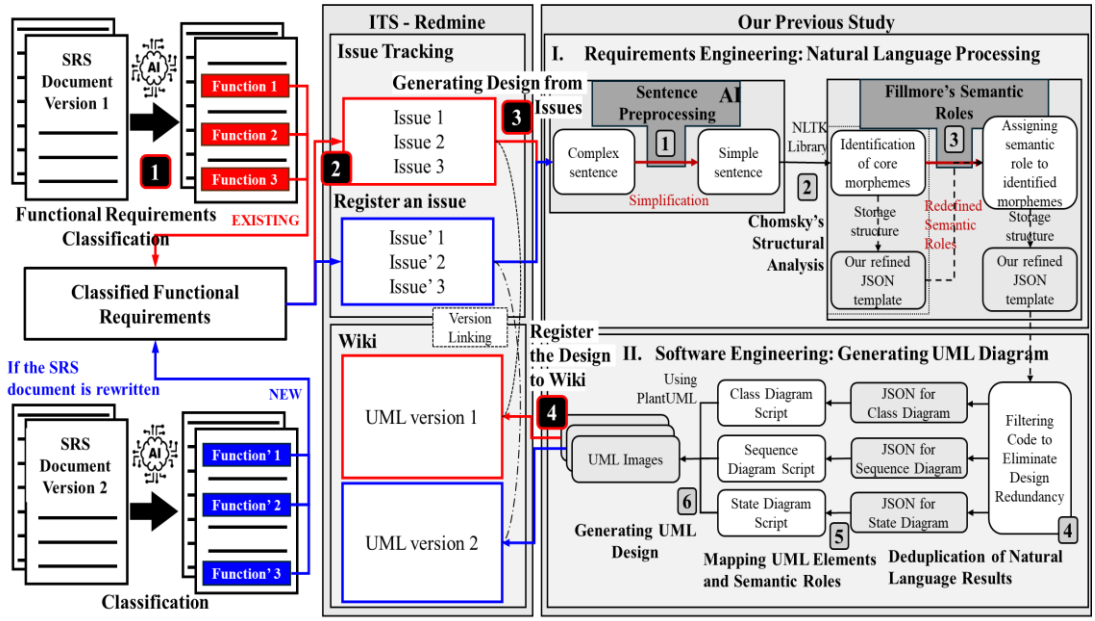

Fig. 1. Automatic Requirements Registration Mechanism Process

The mechanism proposed in this study is as shown in **Fig. 1**.

model to predict the sentence category. The AI model classifies whether a given SRS sentence corresponds to a functional requirement. We use

the PURE (Public Requirements) dataset for learning [11]. The dataset comprises 4,661 requirements, including 2,617 functional requirements and 2,044 non-functional requirements. Rather than analyzing the sentences in detail, we focus on inferring the meaning of each sentence and identifying the category to increase the efficiency of requirements management. This method reduces repetition and the possibility of errors in the existing requirement identification process, which relies on manual work.

3.2 Register an issue in the Issue Tracking System

Once the classification of functional requirements is completed, it must be stored in the Issue Tracking System for project development. We use Redmine among the Issue Tracing Systems. Redmine provides a REST API. Therefore, data can be retrieved quickly and easily in JSON format. Redmine tasks can be categorized by project type. In this study, we configure four Redmine issue types —requirements, design, implementation, and testing — according to the software development process. We read the previously classified functional requirements and automatically register the title and description of the issue in the requirement issue type. After that, developers can check the registered tasks in the Redmine web environment and modify the status, priority, and person in charge of the functions.

3.3 Generate Design from Issues

To extract design from functional requirements, we apply previous research [6]. If requirements have been identified, this step analyzes the requirements in detail. To analyze this systematically, we go through six steps. 1) Requirements written in complex or compound sentence forms are first converted into simple sentences. 2) Sentence structure and morphology are then analyzed using Noam Chomsky's Syntactic Structures Theory [12]. We utilize the Python-based NLTK (Natural Language Toolkit) library, which supports tokenization and part-of-speech tagging. 3) Semantic analysis is performed using Fillmore's Semantic Roles theory [13]. This theory identifies the main verb and finds a relationship between related nouns, assigning a role to each. We utilize GPT to analyze this process. 4) After the sentence analysis, a filtering process is applied to reduce redundancy. Redundancy in requirements can lead to duplication in design and code. 5) The deduplicated semantic roles are then mapped using UML design elements. The resulting design information is stored in a structured format that specifies how the semantic data should be saved. 6) Finally, based on the saved data, we use PlantUML to generate diagram scripts and images.

3.4 Register the Design in the Wiki of the Issue Tracking System

The UML diagram created based on the issue is stored in a repository that Redmine can recognize. The designs outlined in the actual SRS document are categorized under the design type of work, and the design created by this study is stored in the wiki. Redmine's wiki can upload a variety of files, and we use an image file. Additionally, it utilizes a hyperlink to the work corresponding to the design, ensuring trackability between functional requirements and design. Additionally, the wiki can track changes by version. Therefore, when a new design is created, the numbers for each version are given sequentially and can be compared and confirmed by the design of each version.

4. Conclusions

The Software Requirements Specification (SRS) is analyzed manually by developers and registered in an Issue Tracking System for collaborative software development. However, this manual process is time-consuming and can lead to issues such as missing requirements and implementation errors.

Therefore, we propose a mechanism that automatically classifies functional requirements from Software Requirements Specification (SRS) documents. The classified requirements are then integrated into Redmine, an issue-tracking system, where they are registered as issues. This approach minimizes repetitive and error-prone manual tasks, enabling a more efficient collaborative development environment. In addition, automation is extended to the design phase by automatically generating UML models from the classified requirements using natural language. To ensure traceability between requirements and design, the Redmine structure

was configured to align with the study's mechanism, thereby increasing usability.

This study can improve the productivity and accuracy of software development by automating the requirements registration process. Additionally, the process from requirements to design is automated, ensuring traceability at every phase. This enables systematic quality assurance and change management throughout the development process.

We focused on functional requirements; however, future research will include the classification of non-functional requirements and the establishment of a systematic issue registration process. Additionally, we currently utilize the Issue Tracking System to develop the process; however, in the future, we plan to automate the code generation, build, testing, and deployment phases by integrating it with the CI/CD (Continuous Integration/Continuous Deployment) pipeline. Through the integration of automation into the development process, we aim to enhance both the quality and efficiency of software development.

References

[1] D. Dave, A. Celestino, A.S. Varde, and V. Anu, "Management of implicit requirements data in large SRS documents: Taxonomy and techniques," *ACM SIGMOD Record*, vol.51, no.2, pp.18-29, 2022. doi: 10.1145/3552490.3552494

[2] F. Khayashi, B. Jamasb, R. Akbari, and P. Shamsinejadbabaki, "Deep learning methods for software requirement classification: A performance study on the pure dataset," *arXiv preprint arXiv:2211.05286*, 2022. doi: 10.48550/arXiv.2211.05286

[3] K. Rahman, A. Ghani, A. Alzahrani, M.U. Tariq and A.U. Rahman, "Pre-trained model-based NFR classification: Overcoming limited data challenges," *IEEE Access*, vol.11, pp. 81787-81802, 2023. doi: 10.1109/ACCESS.2023.3301725

[4] D. Bertram, A. Dane, S. Greenberg, and R. Walker, "Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams," in *Proc. of the 2010 ACM conference on Computer supported cooperative work*, pp.291-300, 2010. doi: 10.1145/1718918.1718972

[5] T. Merten, B. Mager, P. Hübner, T. Quirchmayr, S. Bürsner, and B. Paech, "Requirements Communication in Issue Tracking Systems in Four Open-Source Projects," in *Proc. of the 6th Int. Workshop on Requirements Prioritization and Communication*, pp.114-125, 2015.

[6] Y.J. Jin, K.D. Kim, D.Y. Yoo, and R.Y.C. Kim, "Adopting Generative AI in Each Phase of Software Life Cycle for Software Development Approach," in *Proc. of the Annual Symposium of KIPS 2025*, pp.401-404, 2025.

[7] J. Cleland-Huang, O.C.Z. Gotel, J.H. Hayes, P. Mader, and A. Zisman, "Software traceability: trends and future directions," in *Proc. of the Future of Software Engineering Proceedings*, pp.55-69, 2014. doi: 10.1145/2593882.2593891

[8] M. Ortu, G. Destefanis, M. Kassab, and M. Marchesi, "Measuring and Understanding the Effectiveness of JIRA Developers Communities," in *Proc. of the 2015 IEEE/ACM 6th International Workshop on Emerging Trends in Software Metrics*, pp.3-10, 2015. doi: 10.1109/WETSoM.2015.10

[9] T. F. Bissyandé, D. Lo, L. Jiang, L. Réveillère, J. Klein, and Y. L. Traon, "Got issues? Who cares about it? A large-scale investigation of issue trackers from GitHub," in *Proc. of the 2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*, pp.188-197, 2013. doi: 10.1109/ISSRE.2013.6698918.

[10] L. Montgomery, C. Lüders, and W. Maalej, "Mining Issue Trackers: Concepts and Techniques," *Handbook on Natural Language Processing for Requirements Engineering*, pp.309-336, 2025. doi: 10.1007/978-3-031-73143-3_11

[11] A. Ferrari, Alessio, G.O. Spagnolo, and S. Gnesi, "Pure: A dataset of public requirements documents," in *Proc. of the 2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp.502-505, 2017. doi: 10.1109/RE.2017.29.

[12] N. Chomsky, Syntactic structures, USA: Mouton de Gruyter, 2002.

[13] C. J. Fillmore, The Case for Case, New York: HR&W, 1968.