Chapter 13
Using Large Language Models to Software Requirements Selection for Scalable, Explainable, and Reliable Results

Mohd Nazim
Department of Computer Science and Engineering, NIET, Greater Noida, India

Shahnawaz Ahmad
Bennett University, India

Mohd Aquib Ansari
https://orcid.org/0000-0002-9083-1523
Galgotias University, India

Arvind Mewada
Bennett University, India

ABSTRACT

The software requirements selection (SRS) is one of the primary activities that decides the success or failure scenarios of software projects. Conventional approaches adopted for the SRS process had several issues, such as bias, limitations of scaling, and absence of clarity. To tackle these limitations, this paper provides a strong integration of the large language models (LLMs) into the SRS process. With the help of the LLMs, it is possible to automate and enhance the process of performing tasks like requirement analysis, requirements prioritization, and decision-making. The

DOI: 10.4018/979-8-3373-4652-6.ch013

proposed LLM-based framework leverages the semantic understanding of LLMs. It analyzes the stakeholders ?(tm) inputs, learns from historical data, and considers existing project constraints to support more precise and efficient requirements handling. The security and explainability concerns of using LLMs in decision-making scenarios are also examined in this paper. Furthermore, the issue of reliability is also addressed to ensure consistency, robustness, and reproducibility of the LLM-driven decisions.

INTRODUCTION

Software Requirements Selection (SRS) plays a significant role in the field of software engineering, as the success or failure of any software project depends on it, (Nazim et al., 2024). It is the mechanism of identifying, evaluating, and prioritizing the software requirements so that the most important, relevant, and feasible requirements can be selected from a large pool of software requirements, to ensure that the final product aligns with stakeholder expectations, business goals, and various technical constraints. It is one of the most critical phases of the software development lifecycle (SDLC) because the improper selection of requirements often leads to project delays, over budget, and even failure of the project, (Nazim, Mohammad, & Sadiq, 2022; Chen & Hwang, 1992; Ji et al., 2023).

The traditional approaches used to SRS mainly rely on manual analysis, stakeholder meetings, and prioritizing frameworks like MoSCoW, Analytic Hierarchy Process (AHP), etc., (Nazim, Mohammad, & Sadiq, 2022; Karlsson, Wohlin, & Regnell, 1998). Despite these techniques being effective in certain contexts, they have several issues as well. One major issue is the manual bias and subjectivity in the stakeholder inputs and decision-making, (Lubos et al., 2024; Berander & Andrews, 2005). The opinions and preferences of different stakeholders may vary. It can affect the fairness and accuracy of the final decision. The complexity in analyzing a large amount of unstructured textual requirements is also a critical challenge, because such requirements are often written in natural language, due to which it hard to understand, compare, and organize them properly, (Jahi & Sami, 2024). Additionally, scalability becomes a major issue when we deal with a large-scale or fast-changing software system. Efficiently managing and analyzing the requirements of any software project becomes tough as the size of the system grows, (Chen, Hu, & Huang, 2025). One more challenge is the difficulty in maintaining traceability and justifying the decisions regarding the selection of software requirements. Without proper documentation and reasoning, it ?(tm)s not easy to know how decisions were made, (Zhang et al., 2006).