Six NFR strategies to improve software performance and security

June 25 2024, by Matt Shipman

Credit: CC0 Public Domain

Non-functional requirements (NFRs) are important aspects of a software system, but are often overlooked by developers because they're not the aspects of a program that users interact with directly. A new study delves

into how developers approach the crafting of NFRs and outlines six best practices for addressing NFRs that can help to ensure a program's security and performance in the near and long term.

NFRs play a meaningful role in how software operates. Examples of NFRs include: how secure a system is; performance, such as how long it takes a system to execute commands from a user; robustness, which refers to how well a system can recover after an error occurs; and maintainability, which is how easy it is for developers to update the system over time.

"NFRs are essential elements of any piece of software," says Wesley K. G. Assunção, co-author of a paper on the topic and an associate professor of computer science at North Carolina State University.

"For this study, we wanted to answer some fundamental questions about NFRs. Who are the experts in charge of these requirements? How do software engineers discuss and manage these types of requirements?"

To that end, the researchers looked at 1,533 pull requests on GitHub pertaining to NFRs, to see how developers discussed and addressed issues related to NFRs. The researchers also performed an in-depth analysis of 63 developers who were particularly active on issues pertaining to NFRs and conducted a survey with 44 developers to get a deeper understanding of their views regarding NFRs.

"Perhaps unsurprisingly, we found that developers discuss NFRs both proactively and reactively—before there's a problem and after there's a problem," says Assunção. "And the developers who were most active in discussing NFRs largely had key roles in the software project. This suggests that NFRs are a primary concern during software maintenance and evolution. The study underscores the role that NFRs play in software quality and the success of software systems."

Based on their analysis and discussions with developers, the researchers identified six key points that are critical to developing and maintaining NFRs that will help a system thrive:

- Prioritization and planning: NFRs should be treated with as much priority as other requirements. They should be planned in advance and reviewed throughout a development project.
- Identification and discussion: NFRs should be identified and discussed early in the development process, ideally in the design phase. During the evolution of the software, these NFRs should be revisited if necessary.
- Use of technologies allied with testing: The adequacy of the NFR can be verified through technologies already approved by the market, where the NFRs associated with those projects satisfy the project's complexity.
- Benchmarks: Using benchmarks to simulate the behavior of a piece of code or algorithm under different conditions is recommended, since it allows developers to review and refactor code when it is not meeting the project-specified NFRs.
- Documentation of best practices: By keeping the NFRs well-documented, developers will have a starting point to address any NFR problem when they appear.
- Long-term mindset: Properly addressing NFRs makes it more likely that a piece of software will have a long lifespan. To guarantee this, a system should have a good user experience, should be designed to scale, and should be easy to maintain by future developers.

"The take-home message here is clear," says Assunção. "We understand that features and functionalities of a software system—the parts of the software that are not NFRs—represent a system's business capabilities and have strategic value for companies. However, our work highlights the fundamental role that NFRs play on the overall quality of a software

system, making them key to a system's success."

The paper, "Understanding Developers' Discussions and Perceptions on Non-Functional Requirements: The Case of the Spring Ecosystem," will be presented July 19 at the 32nd International Conference on the Foundations of Software Engineering, being held in Porto de Galinhas, Brazil.

First author of the paper is Anderson Oliveira of PUC-Rio, Brazil. The paper is co-authored by João Lucas Correia, Juliana Alves Pereira, Daniel Coutinho, Caio Barbosa, Paulo Vítor C. F. Libório and Alessandro Garcia of PUC-Rio; and by Rafael de Mello of the Federal University of Rio de Janeiro, Brazil.

More information: Oliveira et al. Understanding Developers' Discussions and Perceptions on Non-Functional Requirements: The Case of the Spring Ecosystem, DOI: 10.1145/3643750, andersonjso.github.io/preprints/fse24oliveira.pdf

Provided by North Carolina State University