Summary

Companies continuously explore their software systems to acquire evidence for software evolution, such as bugs in the system and new functional or quality requirements. So far, managers have made decisions about software evolution based on evidence gathered from interpreting user feedback and monitoring data collected separately from software in use. These evidence-collection processes are usually unmethodical, lack a systematic guide, and have practical issues. This lack of a systematic approach leaves unexploited opportunities for detecting evidence for system evolution.

The research aims to improve the evidence collection from software in use by understanding useful approaches for gathering user feedback and monitoring data, two important sources of evidence, and combining them. The thesis proposes GESU, a method for gathering evidence from software in use using the design-science research method.

The results show that GESU is not only successful in industrial environments but also yields new evidence for software evolution by bringing user feedback and monitoring data together. This combination helps software practitioners improve their understanding of end-user needs and system drawbacks, ultimately supporting continuous requirements elicitation and product evolution. GESU suggests monitoring a software system based on its goals to filter relevant data (i.e., goal-driven monitoring) and gathering user feedback when the system requests feedback about the software in use (i.e., system-triggered user feedback). The system identifies interesting situations of system use and issues automated requests for user feedback to interpret the evidence from user perspectives. The thesis justified using goal-driven monitoring and system-triggered user feedback with complementary findings of the thesis.

That showed the goals and characteristics of software systems constrain monitoring data. We thus narrowed the monitoring and observational focus on data aligned with goals instead of a massive amount of potentially useless data. Furthermore, we understood that requesting feedback from users using a simple feedback form, relevant to the context of the users’ recent experience, is a useful approach that encourages more users to give feedback. However, this request should not be frequent to disturb users, even though the disturbance may have negligible impacts on the perceived Quality of Experience from the software systems. The findings confirm that goal-driven monitoring and gathering system-triggered feedback of users complement the common approach of continuous monitoring of the whole system and gathering user-triggered feedback, in finding new evidence.

In summary, combining user feedback and monitoring data is helpful to acquire insights into the success of a software system and guide decision-making regarding its evolution. This work can be extended in the future by implementing an adaptive system for gathering evidence from combined user feedback and monitoring data.

Keywords: user Feedback, system monitoring, software evolution, evidence-based software engineering