Week 8: Reproducing Quality Feedback Loop

Nate Bachmeier

TIM-7140:Software Engineering

April 18, 2021

Northcentral University

# Reproducing Quality Feedback Loop

Businesses of all shapes and sizes are leveraging software-based solutions to reduce costs and become more competitive. This vast breadth of use-cases comes with unique constraints that span technical, cultural, and regulatory requirements. For instance, updating an eCommerce website multiple times per day is trivial compared to NASA’s challenges with their Mars Rover. Organizations manage these differences by aligning their software engineering models with the business needs. Regardless of the methodology, all companies desire quality software that gives customers a positive experience. Delivering this outcome is highly challenging and requires multi-discipline skills that span quality assurance and project management.

## Problem Description

SoftTeam (2021) is a software consulting firm with fourteen hundred employees that operate across several European nations. They need a mechanism that promotes agile methodologies, ensures quality, and provides task-level visibility. The business began with a standard scrum implementation with sprints, user stories, and defect backlogs. However, the existing tooling did not provide a single-pane view into the system. Instead, engineers would sprawl project status information across source control (e.g., Git and SVN), task tracking (e.g., Jira), and continuous deployment & integration services (CICD, e.g., Jenkins).

## Study Accomplishments

The Developer Operations Team (DevOps) built Q-Rapid, a dashboarding solution for holistically gaining insights into these systems. Business leaders can monitor the Quality Feedback Loop (QFL) across each engineering stage. For instance, when an engineer commits a revision into source control, it triggers a refresh of code quality metrics (e.g., comment ratios and security warnings). Automation and standard open-source tooling produce most of these metrics, ensuring that information is timely and accurate. Teams can also reference quality requirement documentation such as Mantis defects, Jira user stories, and Open Project timelines.

## Methodology

SoftTeam supports Model-IO, a customer-facing application that converts various software diagrams (e.g., Unified Modeling Language) into Java code. After four years of development, the program contains numerous components with varying quality. The DevOps team chose this complex product as the basis of their case study. This non-trivial development workflow utilizes multiple analytical tools and verifies the management effectiveness.

During the case study, team members would provide feedback regarding the strengths and weaknesses of Q-Rapid. Users can provide feedback through open-ended templates that flow into an issue tracking system. This process sounds relatively ad-hoc, and the researchers exclude quantitative data and sample contributions.

## Contributions

Model-IO’s case study outlines a journey from decentralized software engineering to a central reporting system. The researchers enumerate several challenges and limitations that other implementations might encounter. For instance, onboarding new projects are very tedious, and the custom solution has a considerable initial cost. In exchange, development teams that make that investment are more productive and data-driven. These capabilities improve engineering agility, cost estimate accuracy, and product quality monitoring.

## Extensions and Enhancements

When an organization makes data-driven decisions, it can ensure the right choices for the right reasons (Zambetti, Pinto, & Pezzotta, 2019). However, acquiring those capabilities detract from the company’s mission. Business leaders need to evaluate the system’s resource costs across both humans and machines. If the return on investment is insufficient, then procuring the solution would not make economic sense. Since the publication does not provide any data about those expenses, the reader cannot assess the solution’s net value.

The study also employs an ad-hoc feedback process that allows end-users to request changes. However, there is little insight into those comments, their fix-rate, nor user acceptance data. It is equally plausible that (a) team members love the system or (b) team members find the solution cumbersome. For example, the conclusion specifically notes that adding more tools and projects is tedious. The researchers need to define how that impacts the user’s satisfaction.

# Reproducing the Study

SoftTeam’s journey into data-driven software management is a unique story. Certain aspects generally apply to many organizations, but each business will have a slightly different experience.

## Setup and Resources

Before building a process management system, the organization needs to identify what data is most important. Merely collecting vast piles of statistics increases costs without adding value (Ajis & Baharin, 2019). At NCU-Tech, our leadership team wants insights into a Quality of Service (QoS) model that confirms services are responsive, available, reliable, and secure. They validate these results through infrastructure metrics, synthetic transactions, static analysis tooling, and a custom logging framework. Similar to SoftTeam’s environment, our engineers use GitHub, and product managers leverage Jira.

Aside from the system inputs, there need to be conversations around staffing and purchasing. While it can be initially tempting to save money through a custom solution, the total cost of ownership (TCO) is likely higher than a Commercial-Off-The-Shelf (COTS) product.

## Methodology

After choosing the most critical data sources, the next step is to centralize them into a reporting solution. SoftTeam built this functionality through custom tooling, which is closed-source. Instead, NCU-Tech will reduce risk and initial costs with a COTS application like Splunk or Kibana. This approach frees up resources to focus on value-differentiation through custom extensions versus boiler-plate code. Further, using a COTS solution reduces hiring and training costs, simplifies third-party support contracts, and allows leveraging existing extensions.

There also needs to be official mechanisms for measuring adoption and end-user satisfaction. Software engineers only provide the minimum effort into systems that do not produce meaningful value (Zeller, 2014). System owners can lessen this risk by measuring the time to complete standard tasks. It would also be beneficial to implement a more formal feedback system. For instance, periodically ask quick questions during the login process.

Lastly, the critical objective of Q-Rapid is to improve visibility into software quality management. NCU-Tech needs to create a similar Quality Feedback Loop (QFL) that spans executive leadership, customer support, engineering, and product management. The SoftTeam authors do not detail their engagement solution. One potential solution is to frame the conversation around a handful of goals. For example, new components must have synthetic transactions that cover the core scenarios and use cases. Another example is that all code is maintainable (e.g., according to SonarQube analysis).

## Limitations and Challenges

## Results