Software Quality Understanding by Analysis of Abundant Data (SQUAAD): Towards Better Understanding of Life Cycle Software Qualities

Pooyan Behnamghader and Barry Boehm

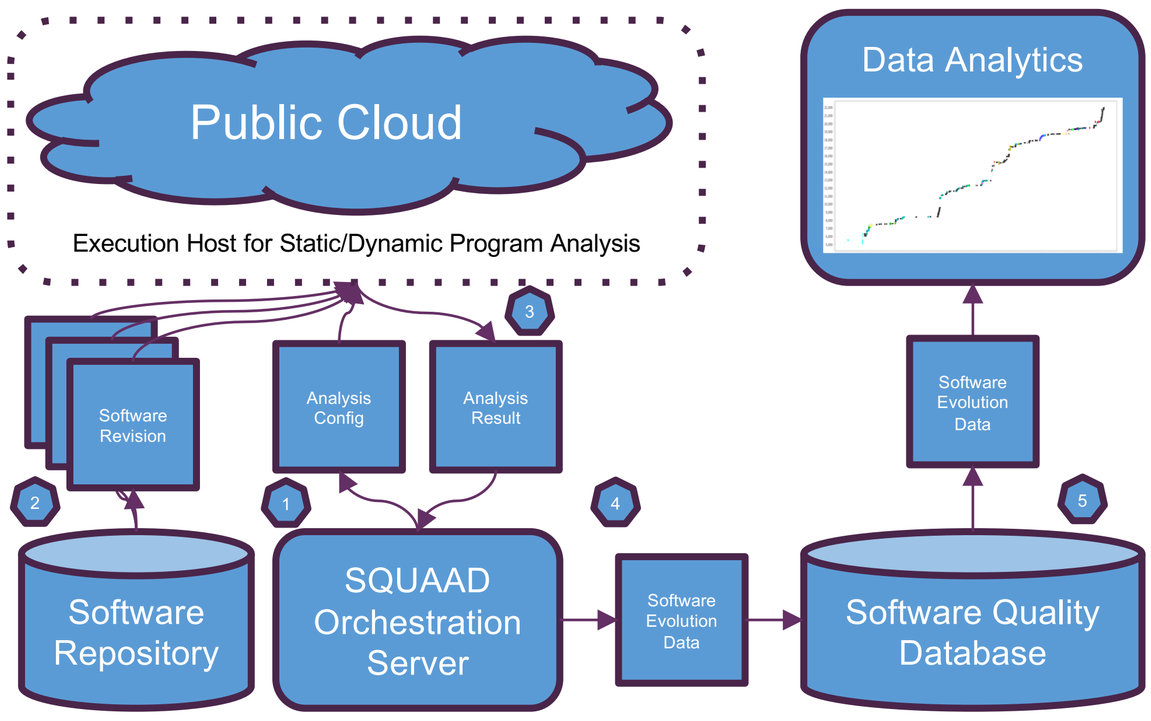
Research Task / Overview

Many ground systems operate for long periods of time, and will have large amounts of software. Even with extensive pre-launch testing, their software will have defects and vulnerabilities that need to be fixed, and further defects and vulnerabilities that emerge as the software evolves. Studies of long-lived software have found that the later the defects and vulnerabilities are found and fixed, the more expensive will be the fix. This phenomenon is now known as technical debt, as the increased cost of making the fix is similar to paying interest on the cost of the fix. Further, delays in living with defective software will have serious impacts on the ground systems' operational capabilities. Thus, investments in methods, processes, and tools for performing large-scale data analytics of their ground system software, or other software of interest will have large returns on investment. A recent study by the Consortium for Software Quality estimated that the cost of technical debt of software worldwide is roughly $2.8 trillion.

Data & Analysis

Experiment's Scale

Org.	Timespan	Sys.	Commits	LOC (MS)
Apache	01/2002-02/2018	38	22627	734
Google	08/2008-01/2018	18	11527	760
Netflix	05/2011-01/2018	12	3684	37
Total	**17 years**	68	37838	**1.5 Billion**



Goals & Objectives

- Enabling large-scale data analysis on large-scale software technical debt.
- Understanding how software and its technical debt evolves.
- Analyzing the impact of each developer and event on software quality.
- Reducing total ownership costs and schedules.

Methodology

An approach to analyze software quality before and after each change.

An automated infrastructure to

- Retrieve a subject system's information from various sources (e.g., commit history and issue repository).
- Distribute hundreds of relevant revisions on multiple cloud instances, efficiently compile each revision, and run static/dynamic programming analysis techniques on it.
- Collect and interpret the artifacts generated by programming analysis techniques to extract quality attributes or calculate change.

A set of statistical analysis techniques tailored for understanding software quality evolution.

- Simple statistics, such as frequency of code smell introduction or correlation between two quality attributes.
- Machine learning techniques, such as clustering developers based on their impact.

Future Research

- Implementing a private cloud solution for close-source applications.
- Extending SQUAAD to include more Static Application Security Testing (SAST) COTS.

Contacts/References

Pooyan Behnamghader, pbehnamg@usc.edu
Barry Boehm, boehm@usc.edu
USC Center for Systems and Software Engineer

B. Boehm and P. Behnamghader. Anticipatory development processes for reducing total ownership costs and schedules. Systems Engineering, 22(5):401–410, 2019.