Software Maintenance

Wednesday 1st of November, 2017

Intro

Maintenance takes up 90% of the lifetime of software.

Software maintenance is the process of modifying software after delivery to correct faults, improve performance or other attributes, or adapt to a change in an environment. A related term is software evolution, which refers to software as it is changed, adapted, etc.

Types of Maintenance

There are 4 types of maintenance:

1. Corrective
    - Removal of bugs
2. Adaptive
    - Adapt the software to a new environment without changing the functionality
3. Perfective / Enhancements
    - Adding new functionality and features
4. Preventive
    - Changes to improve maintainability of the system
    - Trying to pay off a technical debt

Software Entropy

The tendency for software, over time, to become difficult and costly to maintain.

Technical Debt

Financial debt is a temporary shortage of money – debt implies that you'll have the money later but not now.

Technical debt involves having a temporary shortage of time – you take shortcuts to get a software product delivered faster. You borrow time.

Technical debt reflects the cost of additional rework caused by choosing an easy solution now instead of using a better approach that would take longer.

Types

Today technical debt can arguably be found in every step of the design process, not just in the coding.

Lehman's Program Classification

Different types of software evolve in different ways.

S-Type: "Specification"

A program that is correct in the mathematical sense, based on some formal specification.

An S-type program doesn't change due to changing user-requirements. Typically such programs are algorithms, e.g. sin, compression algorithms.

Unsatisfactoriness

It may happen that S-type programs become unsatisfactory.

Revising the specification leads to a new program [...]

Programs don't change – a change is a new program. (Except correcting defects.)

P-Type: "Problem"

A program that solves a problem that can be fully specified, but user's concern is with correctness of results in a specific context.

Example: chess game

- rules of the game can be fully specified

- procedure is to calculate the next best move
- this may not be practically feasible
- have to approximate the solution to be practical/feasible

Needs to give acceptable results, e.g. 90% chance of rain for weather.

E-Type: "Evolving"

A program that operates in, or addresses a problem or activity of the real world.

The program becomes an integral part of the domain within which it operates.

To remain satisfactory to users, it must be continuously evolved.

Most software is E-type.

Changes are intended to improve the user satisfaction.

Lehman's Laws

8 laws of how software evolves, based on observations. The laws only apply to E-type systems.

Law 1: Continuing Change

An E-type system must be continually adapted, else it becomes progressively less satisfactory in use.

Law 2: Increasing Complexity

As an E-type system is changed, its complexity increases and it becomes more difficult to evolve, unless work is done to maintain or reduce the complexity.

Key Challenge in Maintenance

Most software development is bespoke, rather than market-targeted.

Most development is done in projects, where the team is disbanded after the project finishes.

Most maintenance is done by people other than those who wrote the code.