Engineering Design w/Embedded Systems

Lecture 35—Software Maintenance

Patrick Lam
University of Waterloo

April 8, 2013

On Software Maintenance

Who ever heard of a Fourth Year Maintenance Project?

Yet, in the real world, maintenance accounts for much engineer effort.

> ***Software maintenance*** *modifies existing software to fix defects, improve performance, or make the software work in new environments (porting).*

Software Maintenance is Hard

Why?

- must understand the existing code, which can be difficult. (whether code is yours or someone else’s!)
- it’s unglamorous, especially if you’re fixing bugs.
- it’s constrained; you better not break compatibility.

Software Maintenance: Beyond Bug-Fixing

Per T. M. Pigosky[1], approximately 80% of software maintenance activities are unrelated to defect fixes.

Types of maintenance for already-shipped code:

- **Corrective Maintenance**: correct known defects;
- **Adaptive Maintenance**: keep a software product usable in a changing environment;
- **Perfective Maintenance**: improve performance or maintainability; and
- **Preventive Maintenance**: correct latent faults in the product before they manifest themselves.

[1] T.M. Pigosky, *Practical Software Maintenance*, John Wiley & Sons, 1997.

The Problem with Patches

Patching can lead to gnarly code with no design.
Question: What’s the alternative?

Temptation: start over from scratch.

- Sometimes, existing software looks hopeless.
- It’s more fun to redesign rather than maintain.

“Second system effect”: you may do worse by starting over.

Managing Maintenance

Software projects are huge.

Bugs are everywhere, even in shipped software.

10 000s of defects are common.
(Average bug lifetime in Linux: 1.38 years.)

Key to avoiding analysis paralysis: **triage**.

Some bugs are more important than others;

- security fixes—pushed right away;
- minor defects can wait (perhaps forever).

Patch discipline

Changes = potential problems.
Negative progress is always possible.

Before pushing a change,
check that it makes things overall better.

Testing is particularly critical. Also:

- reviews;
- regression tests;
- other verification techniques.

Do less harm than good.