Maintenance

The Silent Killer

Stuart Marks
Core Libraries
Java Platform Group, Oracle

Twitter: @stuartmarks
@DrDeprecator
#J1Maintenance

JavaYourNext (Cloud)

JavaOne ORACLE


© 2010 Stuart W. Marks

Maintenance as a General Phenomenon

- Everything requires maintenance
- Maintenance has costs
- Costs are either paid in cash or are accrued
- Costs are often hidden, thus easily forgotten
- Maintenance is easy to defer
- Deferred maintenance costs can grow superlinearly

"Another flaw in the human character is that everybody wants to build and nobody wants to do maintenance."

— Kurt Vonnegut

Software Doesn't Wear Out

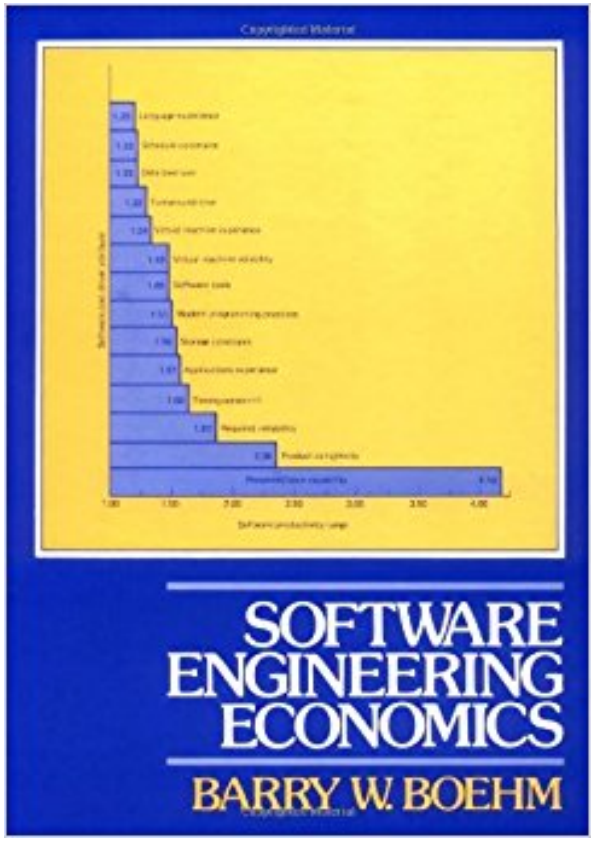
- Software has no moving parts
  - no friction
  - no wear points
  - no stress fatigue
- Therefore, software doesn't need maintenance!
- How many of you believe that?

Sources of “Wear” in Software

- Hardware it runs on can wear out
- The Upgrade Treadmill™
- New feature requests
  - cost of adding new feature
  - evolution may cause introduction of unrelated bugs
- Regulations (GAAP, Sarbanes-Oxley)
- Environment (network config, locale, number of processors)
- Discovery of latent bugs (especially security holes)
- Wear accumulation manifested as technical debt

Boehm, Software Engineering Economics (1981)

- 765 page book
- Chapter on Software Maintenance has 23 pages
- Most software cost estimation material concerns construction
- But majority of software costs incurred after delivery
- Maintenance is 50%-75% of total life cycle costs



Robert L. Glass (2001)

- Maintenance cost is 40%-80% (average 60%) of total software cost
  - on average, maintenance costs more than development
- Software development and maintenance are largely the same activities
  - except for “understand the existing product”
  - that consumes 30% of maintenance time
  - for this reason, maintenance is more difficult than development

Glass, Robert L. “Frequently Forgotten Fundamental Facts About Software Engineering.” IEEE Software V18N3 May/June 2001

JDK Approach to Maintenance

- The JDK is a 21+ year old code base
  - effort divided between new features & maintenance: same engineers!
  - old code occasionally refreshed (warnings cleanup, API & language retrofits)
- Historical & maintainability
  - bug database history, including long-closed bugs
  - source code history, some going back prior to 1.0
  - new features reviewed for maintainability
- Deprecation and removal => reduce maintenance footprint

Open Source Software and Security Vulnerabilities

- "When we are taking in [open source] software, we are taking ownership of it." – David Blevins, CEO TomiTribe
  - that is, ownership of responsibility for maintenance
  - http://blog.sonatype.com/struts2-vulnerabilities-who-is-responsible
- Security company VP: 10% of IT budget goes to keeping SW at current patch levels
- Equifax Breach
  - Struts 2 vulnerability CVE-2017-5638 disclosed and fixed March 2017
  - Equifax was using unpatched Struts 2, was exploited May-July 2017
  - Confidential records of ~143 million consumers were stolen

https://www.wired.com/story/equifax-breach-no-excuse/

Questions for Discussion

- Why is more effort put into planning software construction than maintenance?
- Are there cases when maintenance isn't important?
- What is the design lifetime of the software?
- How can maintenance be made a desirable job role in the project?
- How do we prevent the "race to the bottom"?
- How do you manage OSS or commercial software dependencies?
- If maintenance is important, how can we make sure appropriate attention is paid to it?

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.


JavaOne™
ORACLE®