Software

CS212
Fall 2006

Motivation

- How to build something:
  - Design
  - Develop
  - Analyze
  - Iterate
- To what can you apply this process?

2

Motivation (Continued)

- When do you ask these questions?
  - How do you test it?
  - How do you rate it?
  - How do you market it?
  - How do you maintain it?
  - How do you extend it?
- How do these help the design process?

3

Quality

- Use quality as way to "measure" considerations
- Examples:
  - Is your software a "killer app"?
  - Does your code even work?
  - Better yet, does it even compile?

4

Testing

- Does your software "work"?
  - http://catless.ncl.ac.uk/Risks: Risks to The Public in Computers and Related Systems
- Style issue:
  - Exception-handling vs. good code?
  - How do you determine *all* test cases?
- Do we need testing?

5

Reliability

- Pareto Principle:
  - 80/20 rule applies to "everything"
  - Sturgeon's Revelation?
- NSA study [Drake, IEEE Computer, 1996] on 25 million lines of code:
  - 70-80% of problems were due to 10-15% of modules
  - 90% of all defects in modules containing 13% of code
  - 95% of serious defects from just 2.5% of the code
- "Metrics"?

6

More Reliability

- http://www.cs.utexas.edu/users/ethics/SoftwareStandards/sub/SoftWareReliability.html
- "Software reliability is the probability that software will provide failure-free operation in a fixed environment for a fixed interval of time. Probability of failure is the probability that the software will fail on the next input selected. Software reliability is typically measured per some unit of time, whereas probability of failure is generally time independent…."

7

Validation

- Engineering:
  - Theory: equations and formulas
  - Practice: testing and building
- Software:
  - Formal:
    - Program correctness (and proofs)
    - Classic paper: http://www.spatial.maine.edu/~worboys/processes/hoare%20axiomatic.pdf
  - Informal? Practice?

8

Glass Box Testing

- Tester can access code
- Test all paths
- Can use pareto principle to target likely causes of problems (see profiling)
- More info: http://www.cse.fau.edu/~maria/COURSES/CEN4010-SE/C13/glass.htm

9

Profiling

- Use program help you use pareto principle
  - Example: part of data produced by a profiler (Python), below
- Java:
  - See **`-Xprof`**, **`-Xrunhprof`**
  - http://www.javaperformancetuning.com/news/qotm037.shtml

```
2649853 function calls (2319029 primitive calls) in 53.502 CPU seconds
Ordered by: standard name
 ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   2521    0.227    0.000    1.734    0.001 Drawing.py:102(update)
   7333    0.355    0.000    0.983    0.000 Drawing.py:244(transform)
   4347    0.324    0.000    4.176    0.001 Drawing.py:64(draw)
   3649    0.212    0.000    1.570    0.000 Geometry.py:106(angles)
     56    0.001    0.000    0.001    0.000 Geometry.py:16(__init__)
 343160    9.818    0.000   12.759    0.000 Geometry.py:162(_determinant)
   8579    0.816    0.000   13.928    0.002 Geometry.py:171(cross)
   4279    0.132    0.000    0.447    0.000 Geometry.py:184(transpose)
```

10

Black Box Testing

- Can't see code
  - Why can't you?
  - Why shouldn't you?
- Example techniques:
  - boundary-value analysis: test at extreme values of inputs
  - redundancy: different software for same task: compare them
  - beta testing: throw it to the masses
- Automate?

11

Fuzzer

- The gist:
  - Apply waves of small, random input to program
  - See http://www.cs.wisc.edu/~bart/fuzz
- Fuzzer:
  - Program that generates random data inputs
  - Example: random string inputs into compilers

12