

The Joel Test: 12 Steps to Better Code



<http://www.joelonsoftware.com/articles/fog0000000043.html>

The Joel Test

A score of ≤ 10
means you're in trouble.

CSE 403
requires?

- ✓ 1. Do you use source control?
- ✓ 2. Can you make a build in one step?
- ✓ 3. Do you make daily builds?
- ✓ 4. Do you have a bug database?
5. Do you fix bugs before writing new code?
- ✓ 6. Do you have an up-to-date schedule?
- ✓ 7. Do you have a spec?
8. Do you have quiet working conditions?
9. Do you use the best tools money can buy?
- ✓ 10. Do you have testers as part of the team?
11. Do you have interview candidates write code?
- ✓ 12. Do you do hallway usability testing?

Do you use source control?

- What are the benefits?
 - Allows multiple developers
 - Keep project in consistent state
 - Track changes and enable roll-back
 - Manage multiple versions
 - Save data in case of a disaster
 - Authoritative source for “daily build”

Do you have a one-step build?

- A single script that
 - [does a full checkout from scratch]
 - rebuilds every line of code
 - makes the binary executable files in all versions, languages and #ifdef combinations
 - [creates the installation package]
 - [creates the final media - CDRom, web site, ...]
- All steps are *automated* and exercised regularly
- So, why is this valuable?

Do you do a daily build and test?

- Build the entire product every day and run a good test suite against the new version
 - build from checked in sources
 - automatic and frequent
- Goal: discover problems early, and fix them before disaster strikes
- Benefits
 - Minimizes integration risk
 - Reduces risk of low quality
 - Supports easier defect diagnosis
 - Improves morale - developers, managers, customers

Do you use a bug database?

- You can't keep the bug list in your head
 - Especially with multiple developers and multiple customers

Moreover, looking at the history of bugs can be insightful!

- To characterize a bug consider:
 - how to reproduce it
 - expected behavior, actual behavior
 - responsible party, status, priority
- Best to use what is integrated with your code hosting
- Alternatives: JIRA, Trac, Bugzilla, text file

Do you fix bugs before writing new code?

Why not fix them later?

- Familiar with the code now
- Harder to find (and fix) later
- Later code may depend on this code (try building on quicksand...)
- Bugs may reveal fundamental problems
- Leaving all bugs to the end will make it harder to understand and keep the schedule

Do you have an up-to-date schedule?

- Keeps expectations realistic
 - For the team, customers, stakeholders
- Allows for more accuracy
 - Use experience to improve estimates
- Helps prevent feature creep
 - Don't take on anything without checking the schedule first

Do you have a spec?

- Easier to fix problems at the design stage
- You know what you are trying to build
 - So do your teammates and customer
- More likely that you build the right thing
 - Pieces fit together
 - Customer is satisfied
- Conceptual integrity for your project
- The manual has similar benefits, & is part of the spec
- Undocumented code has no commercial value
 - Joel's example: Netscape Navigator
 - Other examples: Viaweb, Vanu

Do you do hallway usability testing?

Grab someone in the hallway and make them use your code

- Key idea: get feedback *fast*
- A little feedback now >> lots of feedback later
- You will get most of the valuable feedback from the first few users

Joel's Disclaimer

- These are not the only factors that determine success or failure
 - A great team will not help if you are building a product no one wants
 - An incredibly talented team might produce an incredible product without these guidelines
- But all things being equal, these factors indicate a disciplined team that can consistently deliver

Other advice

