

The Joel Test

- Do you use source control?
- Can you make a build in one step?
- Do you make daily builds?
- Do you have a bug database?
- Do you fix bugs before writing new code?
- Do you have an up-to-date schedule?
- Do you have a spec?
- Do programmers have quiet working conditions?
- Do you use the best tools money can buy?
- Do you have testers?
- Do new candidates write code during their interview?
- Do you do hallway usability testing?

"A score of 12 is perfect, 11 is tolerable, but 10 or lower and you've got serious problems. The truth is that most software organisations are running with a score of 2 or 3, and they need serious help, because companies like Microsoft run at 12 full-time."

Joel Spolsky

www.joelonsoftware.com

1. Do you use source control?

Easier working together for programmers; more shared knowledge of what each programmer has done; less chance of code loss; simple examples such as CVS (www.cvshome.org) are free.

In practice it can be *the* major way to ramp up a company's software production.

2. Can you make a build in one step?

Less chance for errors or out-of-synch errors. For example: "On good teams, there's a single script you can run that does a full checkout from scratch, rebuilds every line of code, makes the EXEs, in all their various versions, languages, and #ifdef combinations, creates the installation package, and creates the final media -- CDROM layout, download website, whatever."

3. Do you make daily builds?

Accidentally breaking a build happens often, when not all new files are checked back in for example. This then prevents further work on the build. A daily build checks for this and allows extra confidence in the process. (Common at MS)

4. Do you have a bug database?

A minimal useful bug database must include the following data for every bug:

- complete steps to reproduce the bug
- expected behaviour
- observed (buggy) behaviour
- who it's assigned to
- whether it has been fixed or not

You can use many tools to help or a simple spreadsheet, best if it is openly visible within the team e.g. on a secure intranet.

5. Do you fix bugs before writing new code?

A zero defects method:

means that at any given time, the highest priority is to eliminate bugs *before* writing any new code.

So that errors are not compounded, made more and more expensive to fix the longer they stay in the code

Also it is easier to predict how long it will take to write new code than to fix a mysterious bug.

Finally it could be viewed as being *ready to ship* at any time.

Can also be thought of as evolving more and more featured versions of the working system - a very good model for final year projects.

6. Do you have an up-to-date schedule?

Needs to be there to help with all product planning, launch marketing etc.

Also it allows priority lists to be made, what will be in what is not etc. - helps prevent featuritis or requirements creep.

Good tip for project as well that the schedule should go down to days not months, a simple approach will be discussed later as well as more elaborate methods.

7. Do you have a spec?

Avoided by coders at all cost but at great cost as it means that there is no overall map of the project.

No code without spec and more chances given to write and review specs are needed. Cf CA465 !

8. Do programmers have quiet working conditions?

About valuing people's work and environment and preventing interruptions that prevent programmers especially getting good productive work done.

Anecdotally one of the reasons for high overtime (unpaid) and late nights is because programmers are trying to get some quiet time.

9. Do you use the best tools money can buy?

High-powered workstations allow quicker more hassle free coding; editing GUI fixing etc; relative to salaries, hiring, rent etc. it's a cheap aspect to get right. It also gives an edge in recruitment.

10. Do you have testers?

It is much more productive to have dedicated testing within the company. A strong quality department should have power to veto a release.

Good programmers don't make good testers so getting, retaining and building up a good test department is hard work but worth it.

We will deal in more detail with testing later suffice it to say that it is a core part of a good product team yet is much maligned.

11. Do new candidates write code during their interview?

Getting some practical work done tells a lot more than trick questions at interview. A simple program like string inversion shows how a programmer thinks out problems !

Hiring is a very expensive business so being accurate is important. From your point of view it means that your overall ability to do new things is more important than knowing the tricks in the newest version of java.

12. Do you do hallway usability testing?

Defined as getting the next person who passes the office to use the code to see how they get on. You probably know about Nielsen's ideas that you only need to show the product to 5 or 6 people to get maximum return at UI level

www.useit.com/alertbox/20000319.html