CSE 403
Software Engineering

The Joel Test

What is the Joel Test?

The Joel Test is:

- a checklist of 12 best practices good software teams do
- written in a blog 20 years ago
- by Joel Spolsky (creator of StackOverflow and Trello).

links:
https://www.joelonsoftware.com/2000/08/09/the-joel-test-12-steps-to-better-code/
https://dev.to/checkgit/the-joel-test-20-years-later-1kjk

What is the Joel Test?

The Joel Test is:
- a checklist of 12 best practices good software teams do
- written in a blog 20 years ago
- by Joel Spolsky (creator of StackOverflow and Trello).

12 ✓-or-✗ questions

- 12 ✓ is good,
- 11 ✓ is ok,
- 10 ✓ (or fewer) is "bad".

Today

- Overview of the 12 best practices (briefly).

- The Joel Test 20+ years later.

- Go through made-up software teams/companies
  and see how these best practices play out in the real world.

- Discussion/vote on which team/company has the best practices.

The Joel Test: what's on the list?

1. Do you use source control?
2. Can you make a build [+ release] 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 programmers have quiet working conditions?
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during their interview?
12. Do you do hallway usability testing?

The Joel Test: what's on the list?

1. Do you use source control?
2. Can you make a build [+ release] 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 programmers have quiet working conditions?
9. Do you use the best tools money can buy?
10. Do you have **testers**?
11. Do new candidates write code during their interview?
12. Do you do hallway **usability testing**?

The Joel Test: 20 years later

1. Do you use **source control**?
2. Can you make a **build [+ release] in one step**?
3. ~~Do you make daily builds?~~
   Do you **use CI (clean main branch)**?
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 programmers have **quiet working conditions**?
9. Do you **use the best tools** money can buy?
10. ~~Do you have testers?~~
    Do you do **automated testing** AND do you **have testers**?
11. Do new candidates **write code during** their **interview**?
12. Do you do **hallway usability testing**?

The Joel Test: how does CSE 403 stack up?

403 requires

✓ 1. Do you use **source control**?

✓ 2. Can you make a **build [+ release] in one step**?

✓ 3. ~~Do you make daily builds?~~

   Do you **use CI (clean main branch)**?

✓ 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 programmers have **quiet working conditions**?

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

✓ 10. ~~Do you have testers?~~

    Do you do **automated testing** AND do you **have testers**?

11. Do new candidates **write code during** their **interview**?

✓ 12. Do you do **hallway usability testing**?

The Joel Test: how does CSE 403 stack up?

403 requires

✓ 1. Do you use **source control**?

✓ 2. Can you make a **build [+ release] in one step**?

✓ 3. ~~Do you make daily builds?~~

Do you **use CI (clean main branch)**?

✓ 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 programmers have **quiet working conditions**?

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

✓ 10. ~~Do you have testers?~~

Do you do **automated testing** AND do you **have testers**?

*✓ 11. Do new candidates **write code during** their **interview**?

✓ 12. Do you do **hallway usability testing**?

The Joel Test: how does CSE 403 stack up?

403 requires

✓   1.   Do you use **source control**?

✓   2.   Can you make a **build [+ release] in one step**?

✓   3.   ~~Do you make daily builds?~~

                Do you **use CI (clean main branch)**?

✓   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 programmers have **quiet working conditions?

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

✓   10.   ~~Do you have testers?~~

                Do you do **automated testing** AND do you **have testers**?

*✓   11.   Do new candidates **write code during** their **interview**?

✓   12.   Do you do **hallway usability testing**?

Examples + two disclaimers

1. Plausible companies that we made up, in some cases based on experience.

2. Only some rules are highlighted (assume others are typical/unknown)

The Joel Test

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 programmers have quiet working conditions?
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during their interview?
12. Do you do hallway usability testing?

You work on a team at one of the big tech companies.

(1.) Source control : not only do you use source control, your company has its own suite of internal tools for code reviews, etc., increasing productivity a lot.

(2.) No one-step build : you cannot make the build in one step - in fact you have a "build manager" rotation which consumes an engineer's whole week.

(8.) Open floor plan : you have your own desk, thankfully, but it's on a floor with a few dozen desks and it's often a little busy.

(11.) Coding in interviews : coding is the biggest part of your company's notoriously difficult interview process. As a result, not only can you rely on your coworkers to be technically solid, you frequently learn from them.

The Joel Test

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 programmers have quiet working conditions?**
9. Do you use the best tools money can buy?
10. Do you have testers?
11. **Do new candidates write code during their interview?**
12. Do you do hallway usability testing?

You work for a trendy startup working on something to do with AI, or maybe blockchain.
(Your company's name is a two syllable word, and the same backwards as forwards (e.g. "ozo").)

(2.) One-click builds and **(3.) at-least daily builds** : both use standard continuous integration, resulting in little to no time wasted on fixing broken builds.

(5.) Your team doesn't prioritize fixing bugs and regularly **(6.) doesn't stick to a set schedule** : you're frequently meeting with and demoing the product for series A investors, and management will prioritize new feature launches ahead of fixing known bugs.

The Joel Test
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 programmers have quiet working conditions?
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during their interview?
12. Do you do hallway usability testing?

The Startup Incubator team (3/8)

You work for an early-stage tech startup in an incubator. Things move fast around here.

(2.) One-step builds : Your team uses up-to-date continuous integration tools for (Travis CI).

(8.) Loud conditions : You work in an incubator - so you share your cubicle with three other people, and you share your open floor with at least 12 other companies. It can get pretty loud and distracting on a regular basis.

(9.) On a shoestring budget : Everyone works on their own laptop, partially from home, (different OSes, etc), and you mainly avoid paid software – compatibility issues and some wasted time result.

(12.) Hallway usability testing : As a team you're constantly pinging ideas back and forth and demoing new features, to one another and other people in the company. As a result your UI is great, and you tend to only build useful features.

The Joel Test

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 programmers have quiet working conditions?**
9. **Do you use the best tools money can buy?**
10. Do you have testers?
11. Do new candidates write code during their interview?
12. **Do you do hallway usability testing?**

You work as part of the software team for a big non-tech company (like a hospital, a retail store chain, etc.)
You have quarterly deadlines for projects, and generally follow a more traditional business schedule.

(3) No daily builds : you're on quarterly cycles so you don't test the build on any regular schedule.

(7.) Your team works from a spec , and

(8.) has your own offices.

(10) No testers : Your company is not software focused so you don't have dedicated testers - but you *do* have stringent correctness requirements. As a result you have to spend a lot of time manually testing new features.

The Joel Test

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 programmers have quiet working conditions?**
9. Do you use the best tools money can buy?
10. **Do you have testers?**
11. Do new candidates write code during their interview?
12. Do you do hallway usability testing?

You work for a big enterprise software company. You have quarterly scheduled build releases, follow the Waterfall method, all that.

(3.) No daily builds : and every couple of weeks your team gets blocked on the build being broken by some bug a dozen commits ago. You can imagine a lot of time is lost at the whole company this way...

(6.) Up-to-date schedule : thanks to the company's structured releases, your team always knows what to have done, when. Other teams can count on yours to always hit your deadlines.

(7.) There are specs : Your team is careful to write specs.

(8.) Engineers have their own quiet offices.

The Joel Test

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 programmers have quiet working conditions?**
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during their interview?
12. Do you do hallway usability testing?

The Investing Firm team (6/8)

You work for a big bank or investing firm. Your team does in-house modeling and tooling for its investors.

(7.) No spec : leadership is pretty unclear on what they want you to do, and the software engineers hate writing documentation, so you frustratingly spend more time than you'd like working on projects that are ultimately dropped, or dealing with requirement churn.

(8). Quiet work space : everyone has an office.

(9.) Best tools money can buy : you have your own office and nice hardware. Cost is not a barrier to access any software or computing resources.

(11.) Coding interviews : are pretty difficult and thorough so you can rely on your coworkers to write pretty dependable code.

The Joel Test

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 sc
7. **Do you have a spec?**
8. **Do programmers have quiet working conditions?**
9. **Do you use the best tools money can buy?**
10. Do you have testers?
11. **Do new candidates write code during their interview?**
12. Do you do hallway usability testing?

Your team works for a mission driven not-for-profit. You care a lot about the company, really get along with your co-worker, but some of the engineering practices are ... questionable.

(1.) No source control : Your code is saved on the company file system and collisions are a frequent problem. It's virtually impossible for people to simultaneously work on the same code.

(8.) Quiet work conditions : you don't have offices, but your working spaces are fairly quiet, not like the cacophony of an incubator.

(12.) Hallway testing : you also do a good deal of hallway usability testing.

The Joel Test

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 programmers have quiet working conditions?**
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during their interview?
12. **Do you do hallway usability testing?**

The Research Lab team (8/8)

Your team works for a government-contracted research lab. Your engineering tasks encompass things like big-data biology, rocket engine simulations, etc.

(4.) No bug database - Your company's engineering developed to supplement code written by a principal researcher without software training, and not tracking bugs is one result of the lack of formality. You frequently encounter buggy code but have difficulty institutionally learning from any of these mistakes.

(7.) Your team uses specs, which helps give direction to the team's efforts and avoid wasting time, and

(8.) things are pretty quiet - you work in a lab, and there aren't many distractions.

(11.) No coding in interviews - the lab prioritizes other technical skills, so while some of your coworkers are very experienced engineers, others on your team (who write code) are researchers without a lot of programming experience.
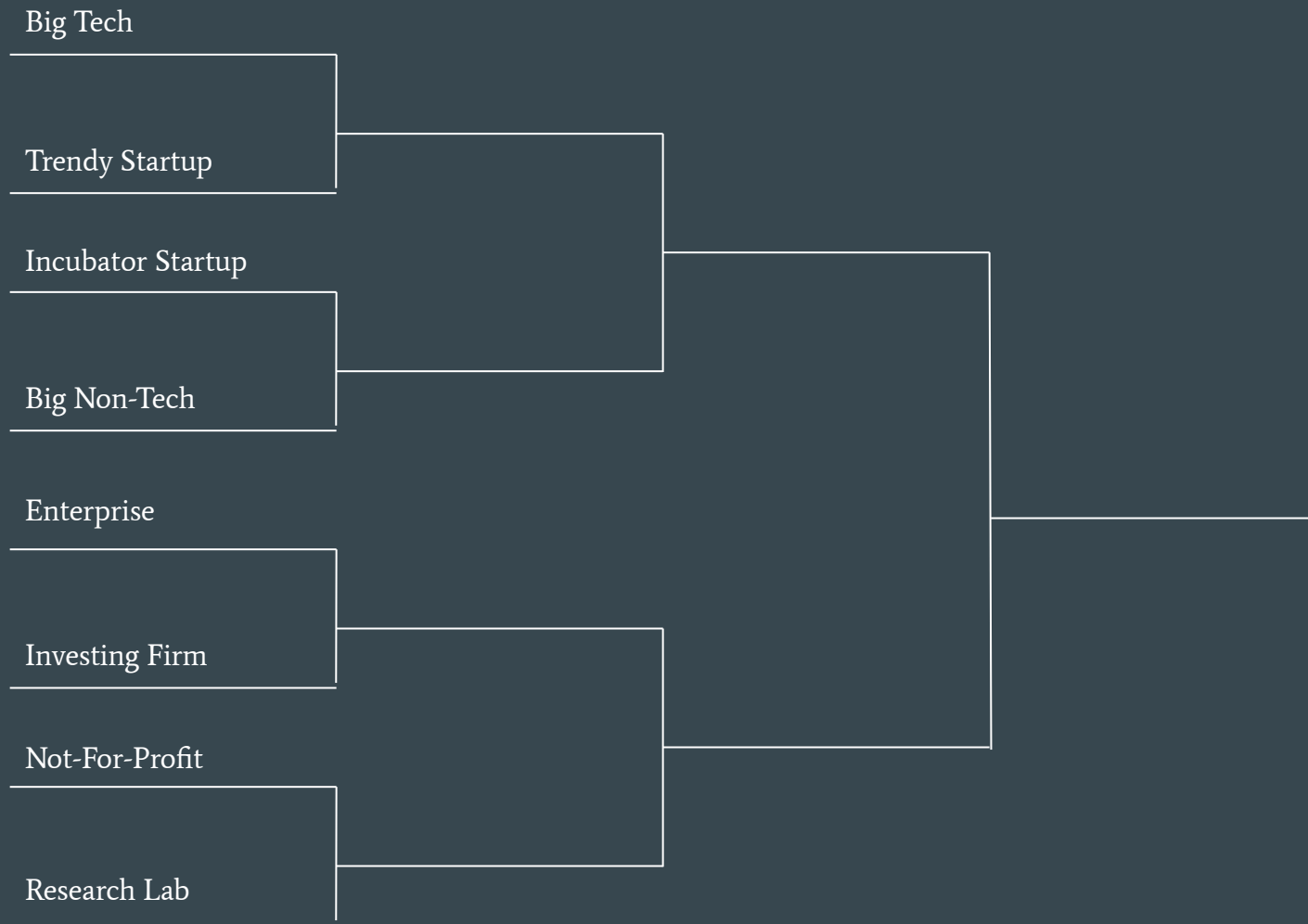
The Joel Test
1. Do you use source control?
2. Can you make a build in one step?
3. Do you make daily builds?
4. **Do you have a bug database?**
5. Do you fix bugs before writing new code?
6. Do you have an up-to-date schedule?
7. **Do you have a spec?**
8. **Do programmers have quiet working conditions?**
9. Do you use the best tools money can buy?
10. Do you have testers?
11. **Do new candidates write code during their interview?**
12. Do you do hallway usability testing?

The Bracket: which team has the best practices?

Big Tech

Trendy Startup

Incubator Startup

Big Non-Tech

Enterprise

Investing Firm

Not-For-Profit

Research Lab

The Bracket: which team has the best practices?

Big Tech

Trendy Startup

Incubator Startup

Big Non-Tech

Enterprise

Investing Firm

Not-For-Profit

Research Lab

The Bracket: which team has the best practices?

Big Tech

Trendy Startup

Incubator Startup

Big Non-Tech

Enterprise

Investing Firm

Not-For-Profit

Research Lab

https://homes.cs.washington.edu/~rjust/courses/CSE403/joel_final.html

Discussion

1. Are these still valid?

   E.g. Google doesn't use testers (10.), has open office spaces (8.), but is considered an expert on best practices.

2. Which are most important and least important?

3. Are any/all of these situational?

The Joel Test
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 programmers have quiet working conditions?
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during their interview?
12. Do you do hallway usability testing?