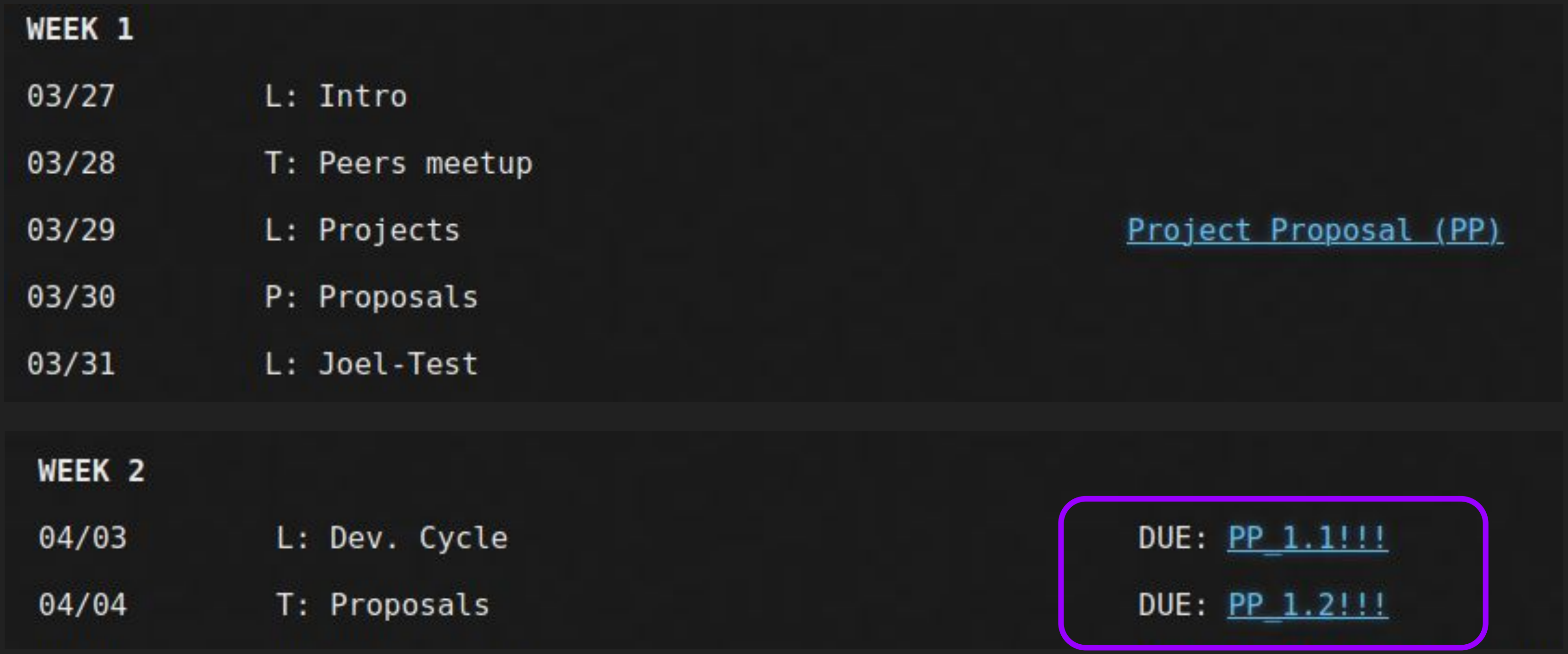
CSE 403

Software Engineering

Spring 2023

#3: The Joel (Spolsky) Test

Logistics: Project Proposal

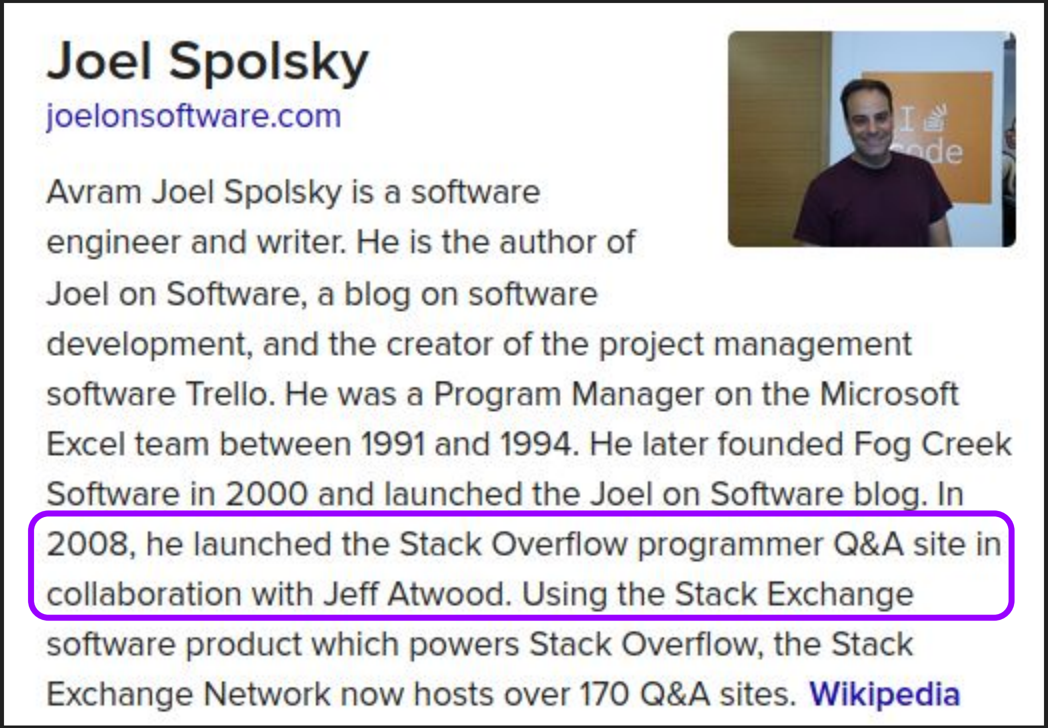


WEEK 1		
03/27	L: Intro	
03/28	T: Peers meetup	
03/29	L: Projects	Project Proposal (PP)
03/30	P: Proposals	
03/31	L: Joel-Test	

WEEK 2		
04/03	L: Dev. Cycle	DUE: PP 1.1!!!
04/04	T: Proposals	DUE: PP 1.2!!!

Any Questions?

Who is Joel?



Joel Spolsky

joelonsoftware.com

Avram Joel Spolsky is a software engineer and writer. He is the author of Joel on Software, a blog on software development, and the creator of the project management software Trello. He was a Program Manager on the Microsoft Excel team between 1991 and 1994. He later founded Fog Creek Software in 2000 and launched the Joel on Software blog. In 2008, he launched the Stack Overflow programmer Q&A site in collaboration with Jeff Atwood. Using the Stack Exchange software product which powers Stack Overflow, the Stack Exchange Network now hosts over 170 Q&A sites. Wikipedia

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".

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

Today

- **Overview of the 12 best practices.**
- Go through made-up software teams/companies
  and see how these best practices play out in the real world.
- Discussion
- Next Steps

The Joel Test: what’s on the list?

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

The Joel Test: what’s on the list?

1. Do you use source control?
2. **Can you make a build [+ release] in one step?**

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?**

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?**

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?**

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?**

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?**

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?**

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?**

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?**

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?**

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?

Question, please!

The Joel Test: what's on the list?

1. Do you
2. Can you step?
3. Do you
4. Do you
5. Do you e?
6. Do you
7. Do you
8. Do prog nditions?
9. Do you ay?
10. Do you
11. Do new eir interview?
12. **Do you**

The Joel Test: what's on the list?

1. Do you
2. Can you step?
3. Do you
4. Do you
5. Do you e?
6. Do you
7. Do you
8. Do prog nditions?
9. Do you y?
10. Do you
11. Do new eir interview?
12. **Do you**

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 (Continuous Integration)**?
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~~ **appropriate working conditions**?
9. Do you **use the best tools** money can buy?
10. ~~Do you have testers?~~
    Do you do **automated testing** AND monitor **test coverage**?
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?

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 (Continuous Integration)**?
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~~ **appropriate working conditions**?
9. ❓ Do you **use the best tools** money can buy?
10. ~~Do you have testers?~~
    ✅ Do you do **automated testing** AND monitor **test coverage**?
11. ❓ Do new candidates **write code during** their **interview**?
12. ❓ Do you do **hallway usability testing**?

Today

- Overview of the 12 best practices.
- **Go through made-up software teams/companies and see how these best practices play out in the real world.**
- Discussion
- Next Steps

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?

Which team has the best practices?

1. Incubator
2. Not-for-profit
3. Big-tech
4. Investing Firm
5. Enterprise
6. Startup
7. Research Lab
8. Big non-tech

Participate: https://forms.gle/iGq7BLHvKqVzxaSv7

The Startup Incubator team (1/8)

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

(2.) One-step builds: Your team uses the GitHub's continuous integration tools.

(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?**

The Not-For-Profit Company team (2/8)

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": Although you have your code in BitBucket, there is not a good process/effort to integrate upgrades from collaborators.

(5.) Lower bug priority: The company has little resources to keep up with new requirements. Bugs are only tackled only when somethings breaks really bad.

(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 Big Tech Company team (3/8)

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?

The Investing Firm team (4/8)

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. In fact, maybe you have too much time away from your team.

(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.

(10.) Do you have testers: Yes, but they are mostly focused on higher level issues, like the results of analysis.

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?

The Enterprise Company team (5/8)

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.

(9.) Best tools available: Not really. Because of the companies' partnerships, you have to stick with the provided tools and it is really hard to try new ones.

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 Trendy Startup team (6/8)

You work for a trendy startup working on something to do with deep learning, 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 Research Lab team (7/8)

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

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

(11.) No coding in interviews - the company prioritizes other technical skills, so while some of your coworkers 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 Big Non-Tech Company team (8/8)

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

(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?

Which team has the best practices?

1. Incubator
2. Not-for-profit
3. Big-tech
4. Investing Firm
5. Enterprise
6. Startup
7. Research Lab
8. Big non-tech

RESULTS!

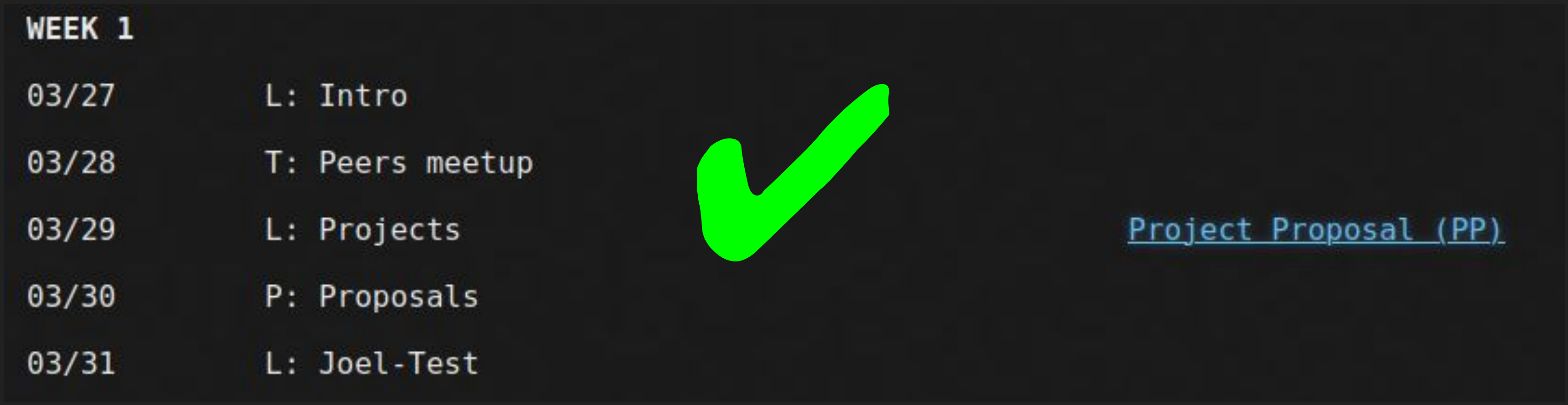
Today

- Overview of the 12 best practices.
- Go through made-up software teams/companies and see how these best practices play out in the real world.
- **Discussion**
- Next Steps

Discussion

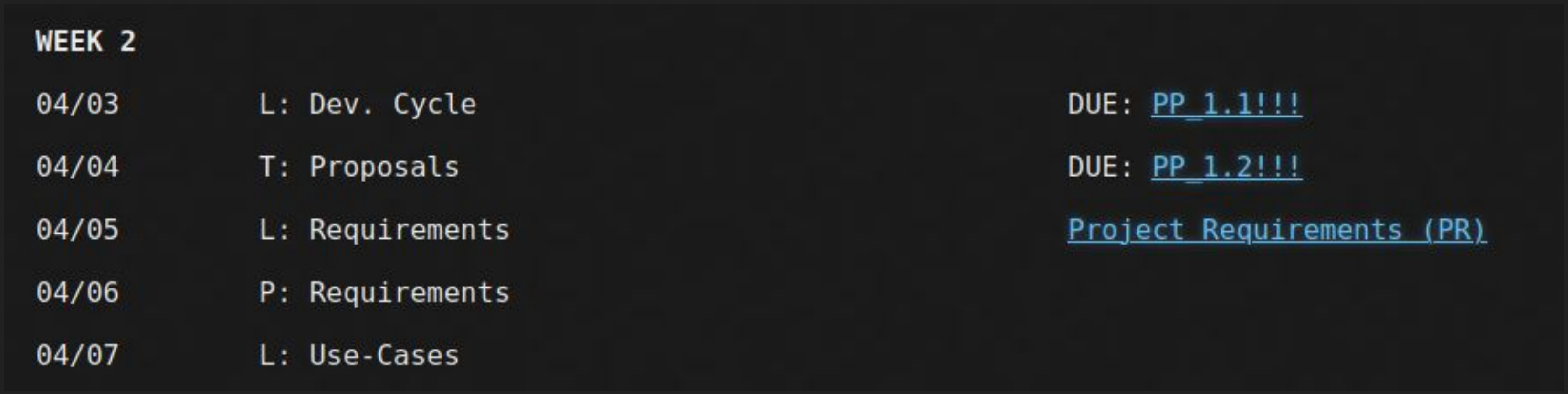
1. Do you use **source control**?
2. Can you make a **build [+ release] in one step**?
3. ~~Do you make daily builds?~~
   Do you **use CI (Continuous Integration)**?
4. Do you have a **bug database**?
5. Do you **fix bugs before** writing **new code**?
6. Do you have an **up-to-date schedule**?
7. Do you **have a spec**?
8. Do programmers have ~~quiet~~ **appropriate working conditions**?
9. Do you **use the best tools** money can buy?
10. ~~Do you have testers?~~
    Do you do **automated testing** AND monitor **test coverage**?
11. Do new candidates **write code during** their **interview**?
12. Do you do **hallway usability testing**?

Next Steps



WEEK 1

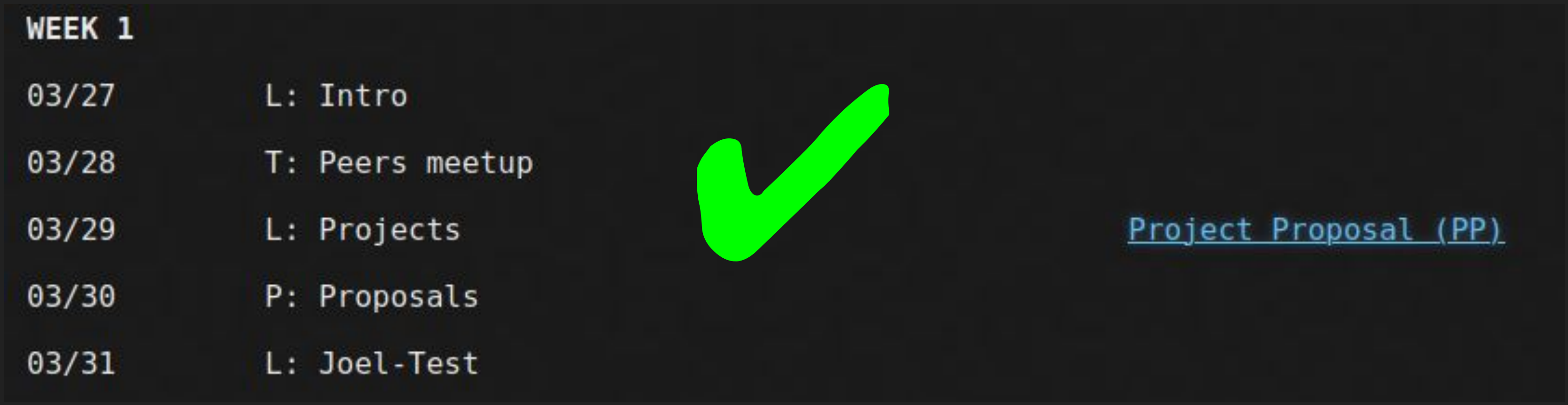
03/27	L: Intro		
03/28	T: Peers meetup		
03/29	L: Projects	✓	Project Proposal (PP)
03/30	P: Proposals		
03/31	L: Joel-Test		



WEEK 2

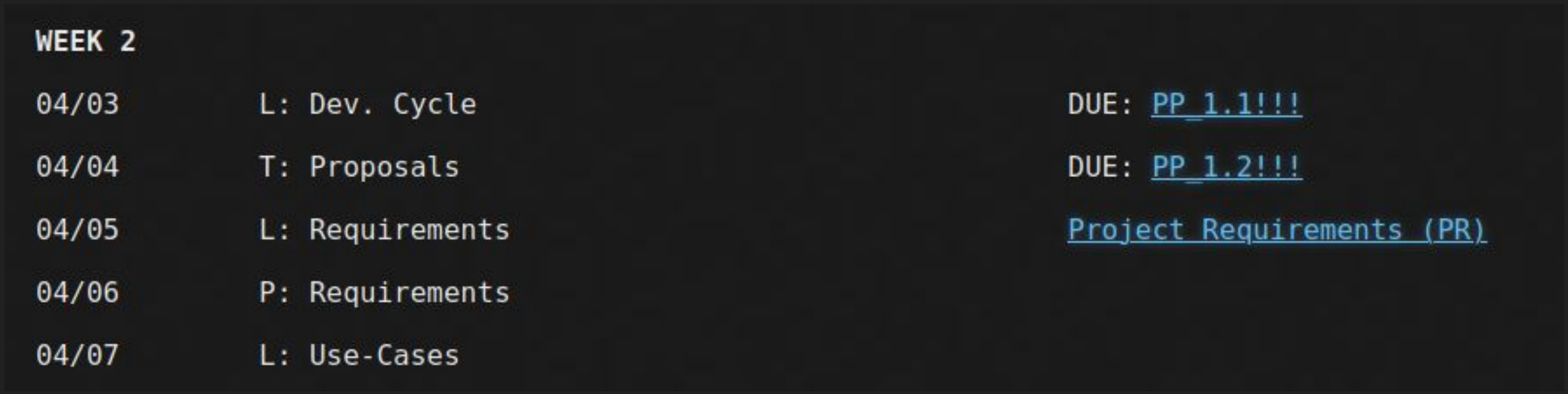
04/03	L: Dev. Cycle	DUE: PP 1.1!!!
04/04	T: Proposals	DUE: PP 1.2!!!
04/05	L: Requirements	Project Requirements (PR)
04/06	P: Requirements	
04/07	L: Use-Cases	

Next Steps



WEEK 1

03/27	L: Intro	
03/28	T: Peers meetup	
03/29	L: Projects	Project Proposal (PP)
03/30	P: Proposals	
03/31	L: Joel-Test	



WEEK 2

04/03	L: Dev. Cycle	DUE: PP_1.1!!!
04/04	T: Proposals	DUE: PP_1.2!!!
04/05	L: Requirements	Project Requirements (PR)
04/06	P: Requirements	
04/07	L: Use-Cases	

Questions, please!