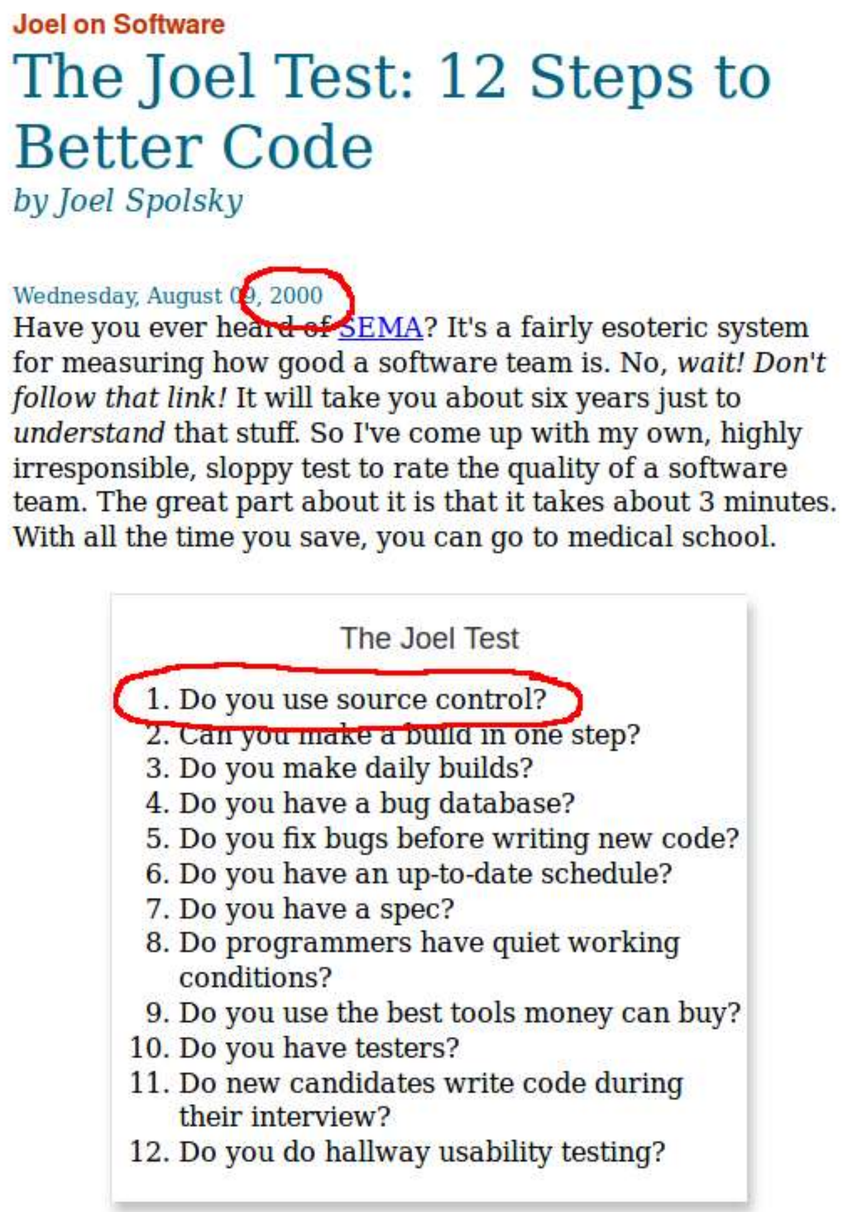
SOURCE CONTROL



Joel on Software

The Joel Test: 12 Steps to Better Code

by Joel Spolsky

Wednesday, August 09, 2000

Have you ever heard of SEMA? It's a fairly esoteric system for measuring how good a software team is. No, *wait! Don't follow that link!* It will take you about six years just to *understand* that stuff. So I've come up with my own, highly irresponsible, sloppy test to rate the quality of a software team. The great part about it is that it takes about 3 minutes. With all the time you save, you can go to medical school.

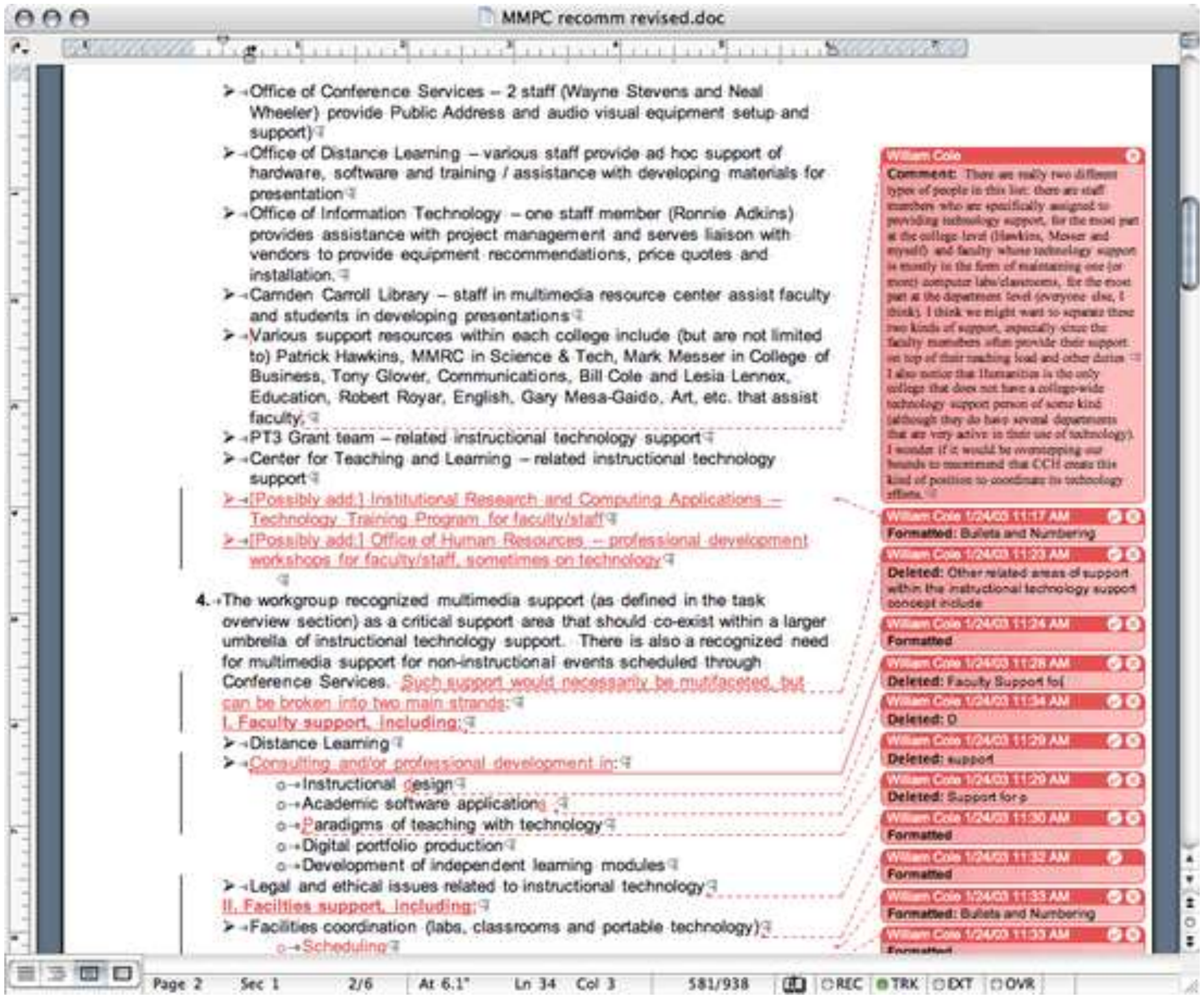
The Joel Test

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?

15 years ago... blog of the guy who would go onto co-found Stack Overflow

1. Do you use source control?

- ...if you don't have source control, you're going to stress out trying to get programmers to work together. Programmers have no way to know what other people did.
- Mistakes can't be rolled back easily. The other neat thing about source control systems is that the source code itself is checked out on every programmer's hard drive -- I've never heard of a project using source control that lost a lot of code.



MMPC recomm revised.doc

- Office of Conference Services – 2 staff (Wayne Stevens and Neal Wheeler) provide Public Address and audio visual equipment setup and support)
- Office of Distance Learning – various staff provide ad hoc support of hardware, software and training / assistance with developing materials for presentation
- Office of Information Technology – one staff member (Ronnie Adkins) provides assistance with project management and serves liaison with vendors to provide equipment recommendations, price quotes and installation.
- Camden Carroll Library – staff in multimedia resource center assist faculty and students in developing presentations
- Various support resources within each college include (but are not limited to) Patrick Hawkins, MMRC in Science & Tech, Mark Messer in College of Business, Tony Glover, Communications, Bill Cole and Lesia Lennex, Education, Robert Royar, English, Gary Mesa-Gaido, Art, etc. that assist faculty.
- PT3 Grant team – related instructional technology support
- Center for Teaching and Learning – related instructional technology support
- [Possibly add:] Institutional Research and Computing Applications – Technology Training Program for faculty/staff
- [Possibly add:] Office of Human Resources – professional development workshops for faculty/staff, sometimes on technology

4. The workgroup recognized multimedia support (as defined in the task overview section) as a critical support area that should co-exist within a larger umbrella of instructional technology support. There is also a recognized need for multimedia support for non-instructional events scheduled through Conference Services. Such support would necessarily be multifaceted, but can be broken into two main strands:

I. Faculty support, including:

- Distance Learning
- Consulting and/or professional development in:
  - o Instructional design
  - o Academic software applications
  - o Paradigms of teaching with technology
  - o Digital portfolio production
  - o Development of independent learning modules
- Legal and ethical issues related to instructional technology

II. Facilities support, including:

- Facilities coordination (labs, classrooms and portable technology)
  - o Scheduling

William Cole
Comment: There are really two different types of people in this list: there are staff members who are specifically assigned to providing technology support, for the most part at the college level (Hawkins, Messer and myself) and faculty whose technology support is mostly in the form of maintaining one (or more) computer labs/classrooms, for the most part at the department level (everyone else, I think). I think we might want to separate these two kinds of support, especially since the faculty members often provide their support on top of their teaching load and other duties.
I also notice that Humanities is the only college that does not have a college-wide technology support person of some kind (although they do have several departments that are very active in their use of technology).
I wonder if it would be overstepping our bounds to recommend that CCH create this kind of position to coordinate its technology efforts.

William Cole 1/24/03 11:17 AM
Formatted: Bullets and Numbering

William Cole 1/24/03 11:23 AM
Deleted: Other related areas of support within the instructional technology support concept include

William Cole 1/24/03 11:24 AM
Formatted

William Cole 1/24/03 11:26 AM
Deleted: Faculty Support fo(

William Cole 1/24/03 11:34 AM
Deleted: D

William Cole 1/24/03 11:29 AM
Deleted: support

William Cole 1/24/03 11:29 AM
Deleted: Support for p

William Cole 1/24/03 11:30 AM
Formatted

William Cole 1/24/03 11:32 AM
Formatted

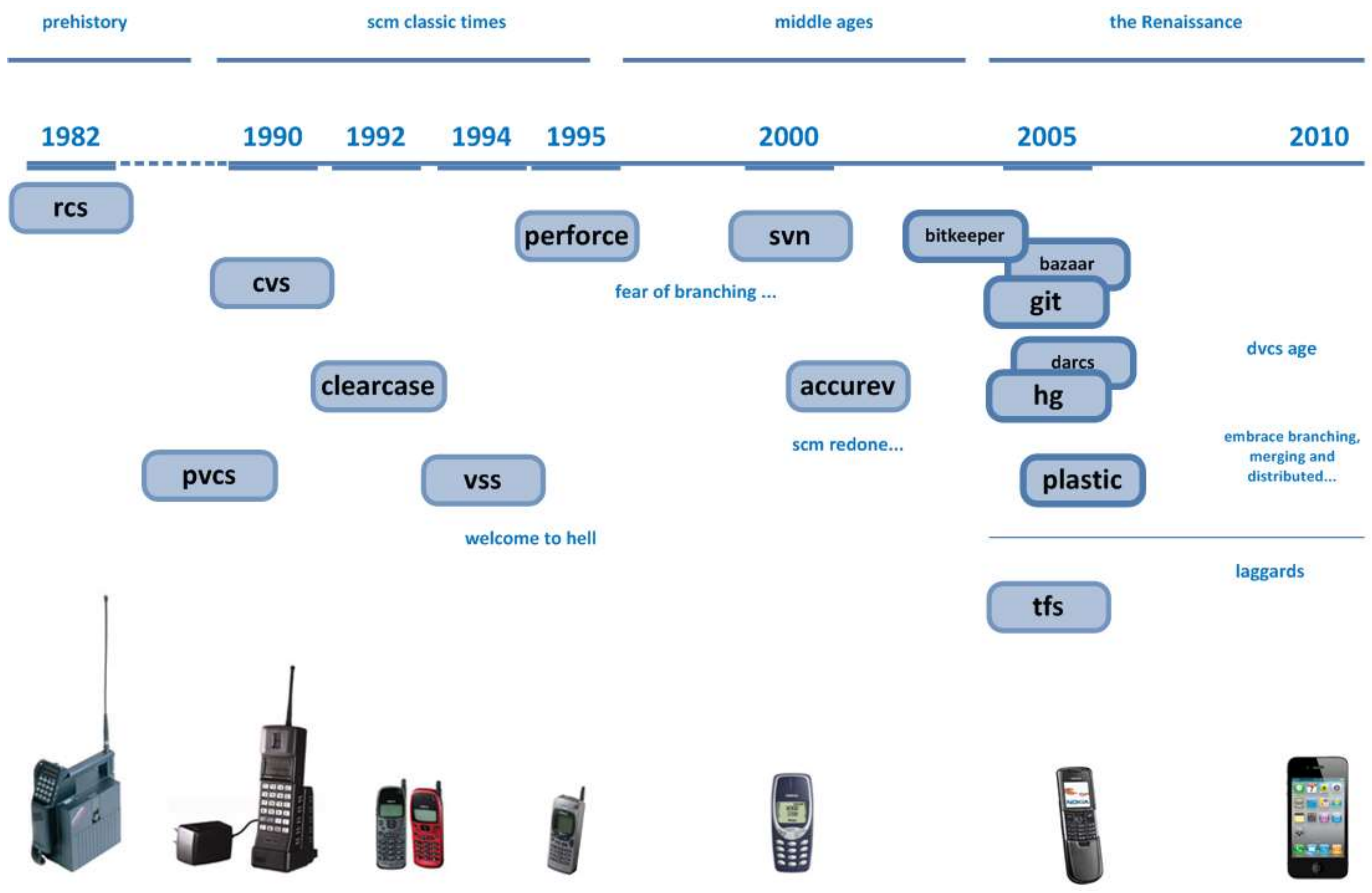
William Cole 1/24/03 11:33 AM
Formatted: Bullets and Numbering

William Cole 1/24/03 11:33 AM
Formatted

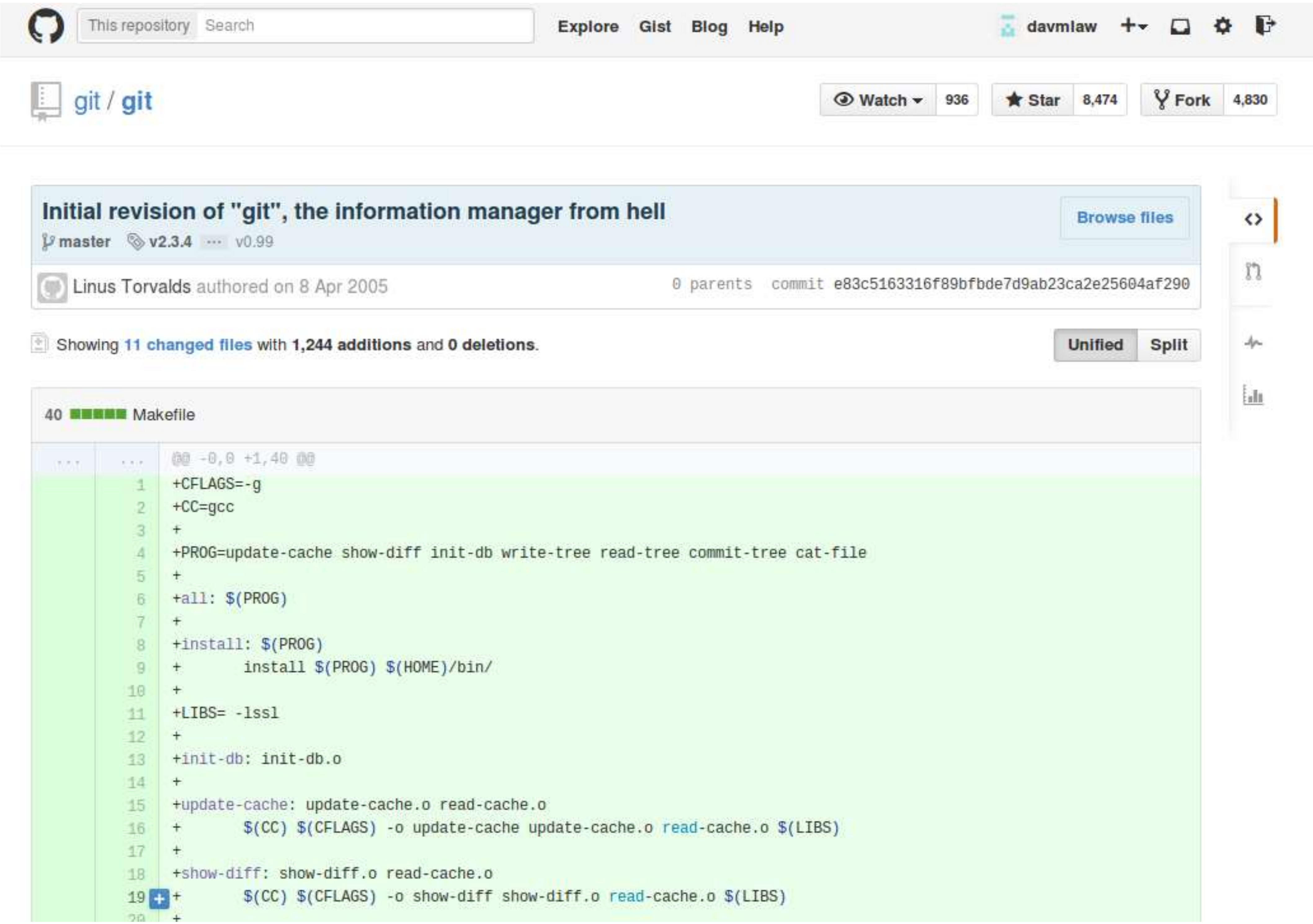
Many people's first introduction to the concept:

MS Word, Track changes - 2004

A history of source control



http://codicesoftware.blogspot.com/2010/11/version-control-timeline.html



Initial revision of "git", the information manager from hell

master v2.3.4 ⋯ v0.99

Browse files

Linus Torvalds authored on 8 Apr 2005   0 parents commit e83c5163316f89bfbde7d9ab23ca2e25604af290

Showing **11 changed files** with **1,244 additions** and **0 deletions**.

40 🟩🟩🟩🟩🟩 Makefile

```
@@ -0,0 +1,40 @@
+CFLAGS=-g
+CC=gcc
+
+PROG=update-cache show-diff init-db write-tree read-tree commit-tree cat-file
+
+all: $(PROG)
+
+install: $(PROG)
+	install $(PROG) $(HOME)/bin/
+
+LIBS= -lssl
+
+init-db: init-db.o
+
+update-cache: update-cache.o read-cache.o
+	$(CC) $(CFLAGS) -o update-cache update-cache.o read-cache.o $(LIBS)
+
+show-diff: show-diff.o read-cache.o
+	$(CC) $(CFLAGS) -o show-diff show-diff.o read-cache.o $(LIBS)
+
```

Summary

- Source Control / Revision Control / Version Control – tools used to help manage and track changes in text files (ie source code)
- Git – a source control tool, written by Linus Torvalds that we will be learning today
- Github - A website, which makes it easy to work collaboratively on git repositories. Bitbucket is similar

Recap

- Use source control
- It has been “best practices” for 15+ years
- Use source control
- Even if it's just you, use source control

3 things to help you understand Git

- Hashing
- Diffs
- Directed Acyclic Graphs

Hashes

- Convert any-length strings into (shorter) fixed sized strings.
- Useful for hash tables, cryptography, data transmission and many other uses.

```
$ echo "Hello world" | md5sum
f0ef7081e1539ac00ef5b761b4fb01b3  -
```

Try:

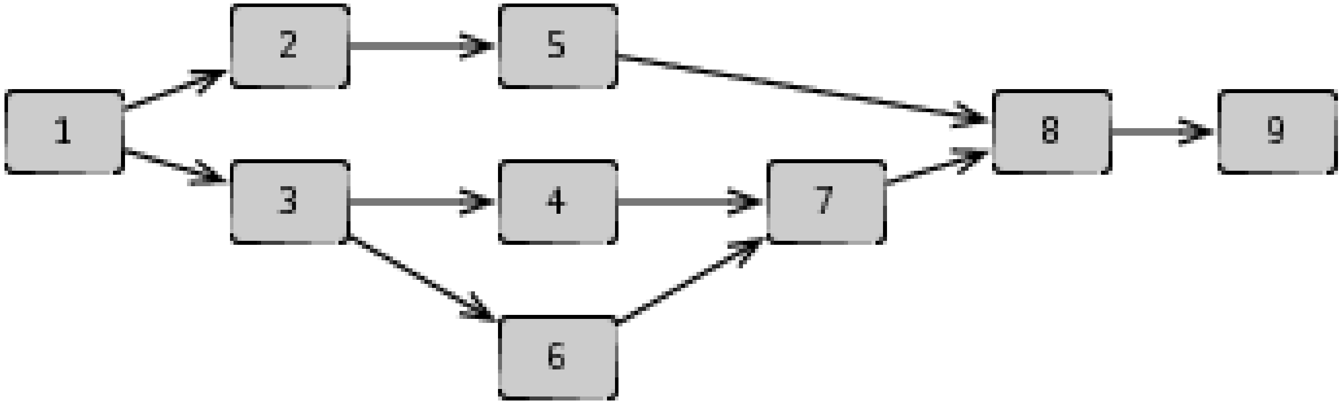
- Copying files, hashing them.
- Creating files that are identical (not via copying), hashing them
- Modify files, hash them. Modify back.
- See how it works?

Diff

- Create a file with lots of lines
- Make a copy
- Modify a line ½ way through
- Run diff file1.txt file2.txt

Directed Acyclic Graph

- Directed (ie the link is always the same direction, ie between parent->child)
- Acyclic – no loops
- Graph – has nodes & edges



Install first...

- Linux:
- sudo apt-get install -y git meld
-

Git Hello World

```
git help
mkdir -p ~/localwork/git_tutorial
cd ~/localwork/git_tutorial
git init
(ls -l .git)
echo “Hello world” > hello.txt
git add hello.txt
git commit -m “initial commit”
```

Git Hello World

git log

Commit contains - Hash, date, message, person/email.

```
(Modify hello.txt)
git add hello.txt
git diff # Shows what changed
git commit -m “modified”
git log
rm hello.txt
git diff
git checkout hello.txt # revert
cat hello.txt
```

Moving through time...

```
git log
git checkout 851ae # Copy your 1st commit hash
cat hello.txt
git checkout master
cat hello.txt
```

Branches (easy fast forward)

```
git branch feature
git checkout feature
(modify hello.txt)
git commit -a -m “modified in feature branch”
git log # has commit in feature branch
git checkout master
git log # doesn't have commit
git merge feature # Merge FROM feature into current (master)
Updating 219ddc3..70d4bfa
Fast-forward
 hello.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
git branch
git branch -d feature # Delete feature
git log --graph
```

Branches (with auto merge)

```
(make hello.txt have lots of lines, and commit)
git branch feature
git checkout feature
(modify hello.txt on top lines)
git commit -a -m “modified in feature branch”
git checkout master
(Modify hello.txt on bottom lines)
git commit -a -m “modified in master”
git merge feature # auto-merges
git log --graph
```

Blame

Who did this??!

```
git blame hello.txt
```

Branches (with manual merge)

```
git branch my_branch
git checkout my_branch
(modify hello.txt)
git commit -a -m “modified in my_branch”
git checkout master
(Modify hello.txt on SAME lines)
git commit -a -m “modified in master”
git merge my_branch # can't do it
cat hello.txt
git mergetool # Visual tool to do it
git log --graph
```

Remote repositories

- Clone
- Pull
- Push
- If a remote won't accept my push, 99% of the time pull, merge, commit, push will fix it.

Git ecosystem

- Github
- Bitbucket
- SmartGit (GUI)