Software Development – Basics

- Not talking about ***programs***, but about ***program system products***

- Know about programs, what about PSPs?

Software Development – Basics

- Program: complete by itself
- Ready to run:
  - By author only
  - For planned inputs only
  - On system on which it was developed

Program System

- Each program is a component in integrated collection
- Precisely defined interface: all programs comply
- Each program sticks to reasonable resources
- Each program is tested with other programs, many combinations

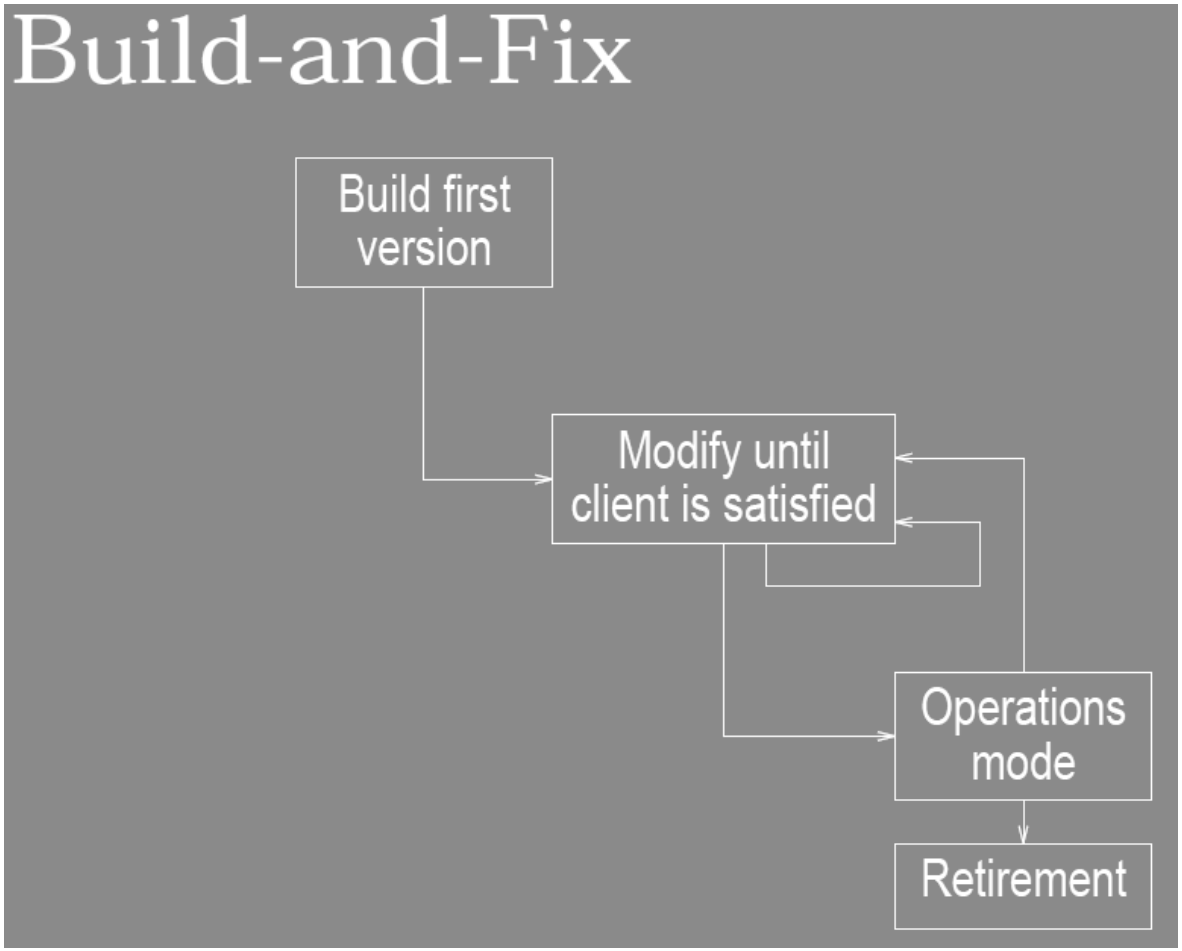
Program Product

- Product can be run, tested, repaired, extended by anyone
- Runs on multiple platforms
- Accommodates many sets of data
- Generic range and form of input
- Provides response to invalid inputs
- Provides documentation for users and developers

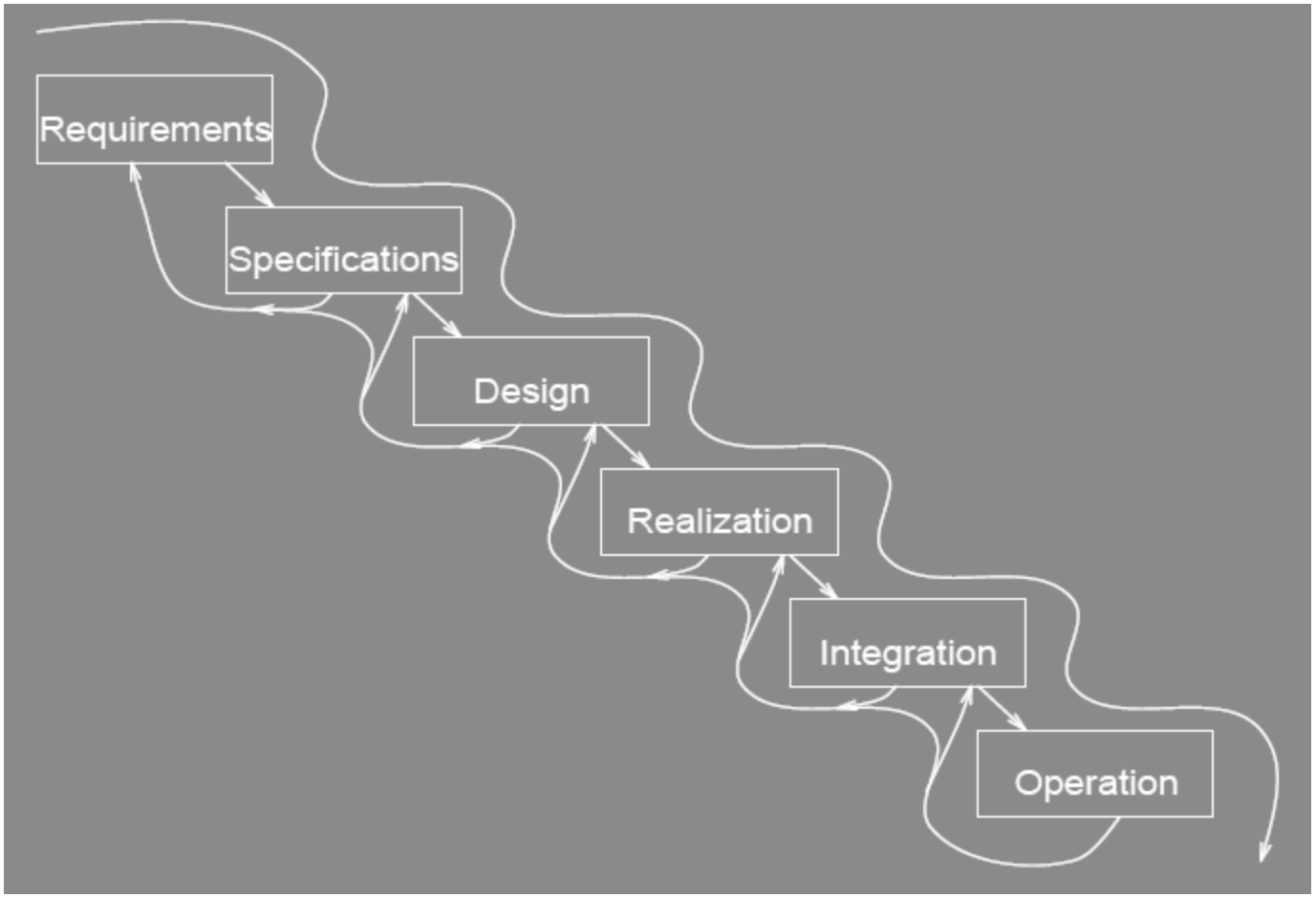
Program Product

- We think about programs, instead of PSPs.
- Affects all calculations:
  - Ease vs. difficulties
  - Time
  - Costs
- All off by a magnitude

Common SPM models: Build-N-Fix

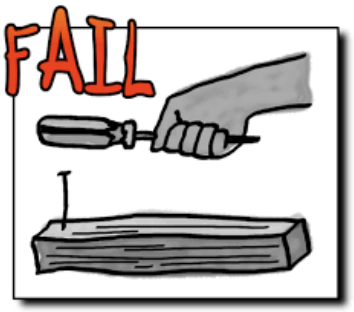


Common SPM models: Waterfall cycle



A software developers toolkit: Which ones do you have?

- **IDE and debugger**
- **Compare files and find differences**
- **Work with regular expressions**
- **Source control**
- **Versioning and sharing**
- **Software testing**



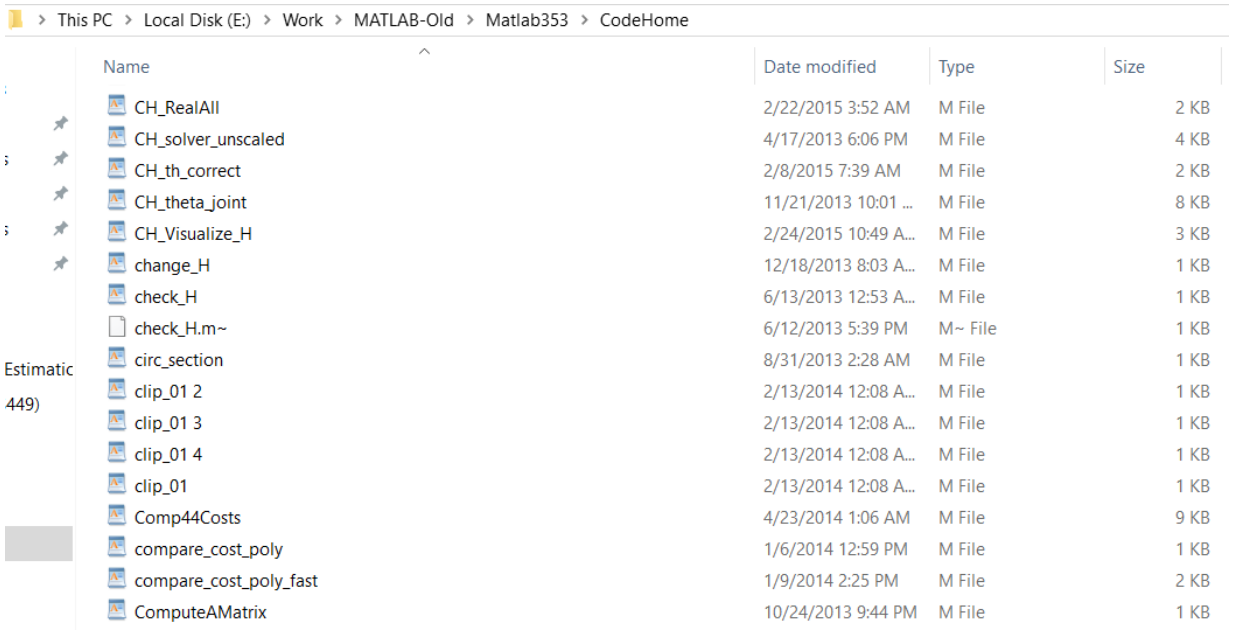
Visual Studio

TortoiseSVN

DIFFMERGE

expresso

I know you too have this on your drive



A dump of code

What to do with it? Close this window and NEVER come back

US

A slightly better approach for beginners

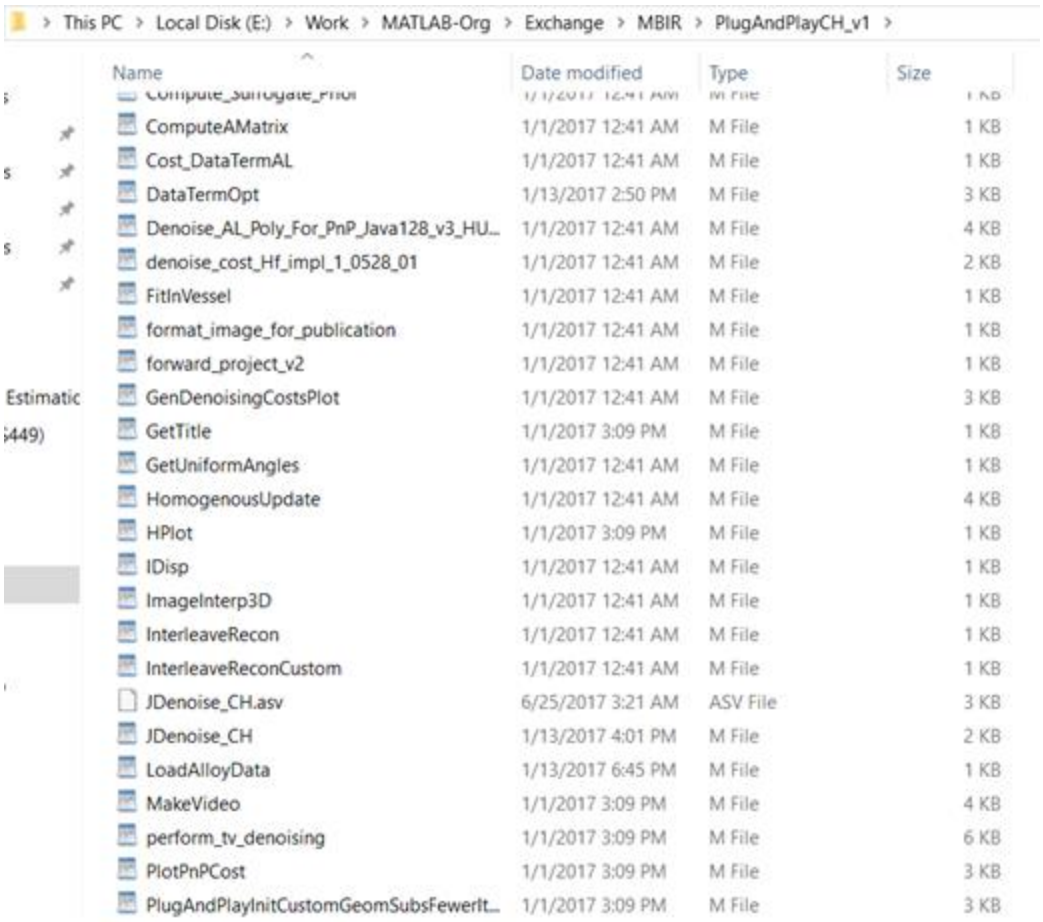
- “I know I should comment, write good code etc. *but I didnt*.”
- **How to salvage something out of this dumpster?**

Here is your way backwards

- When your project ran, always save that version as _EXE(even when its due tomorrow)

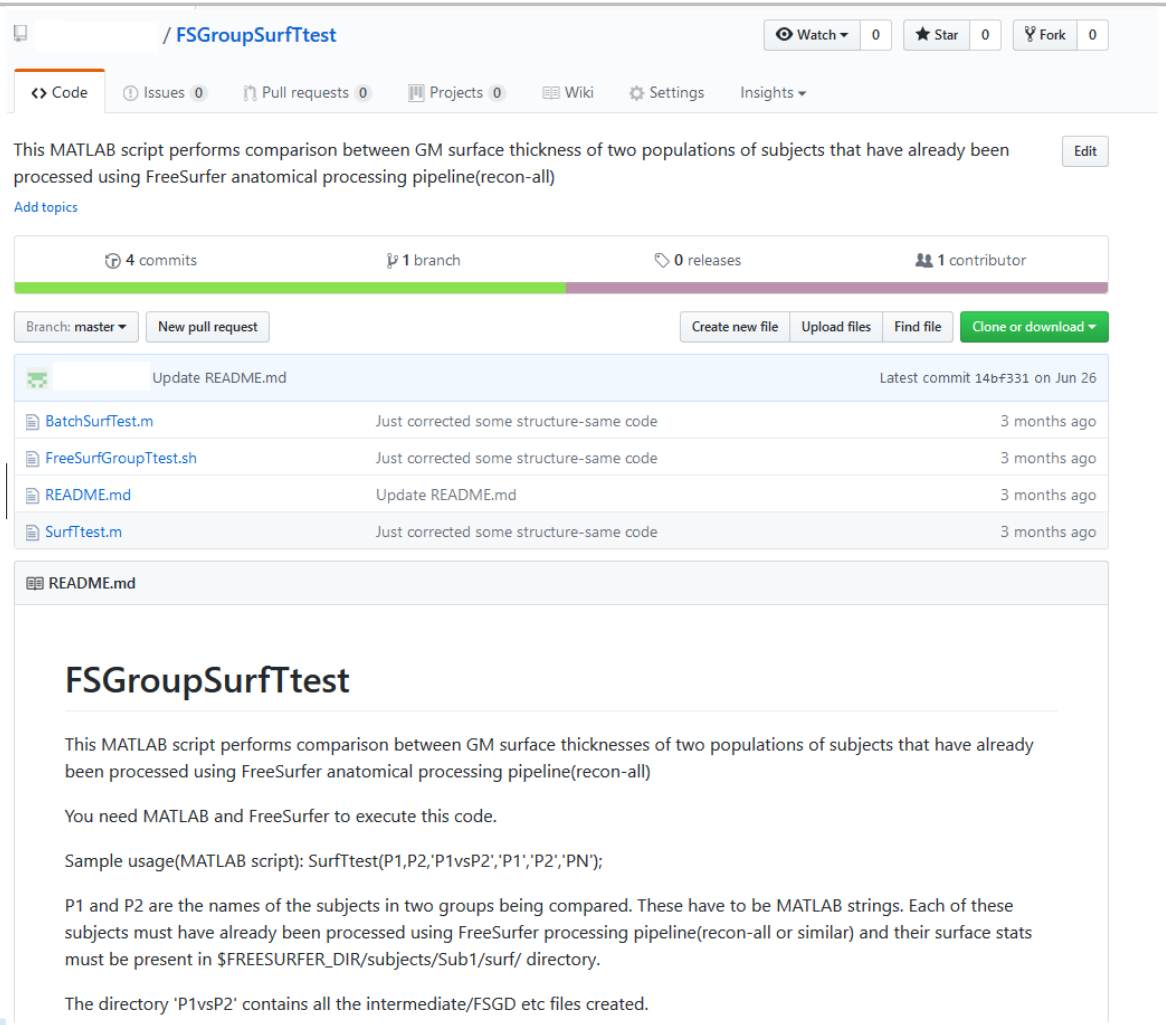
- After you are done with submission etc., go to _EXE file
- Use a dependency tracker to find all related code, then copy it to a new folder
- Add documentation, and check out on Git.

Looks better



US

Much better



/ FSGroupSurfTtest

Watch 0 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Settings | Insights

This MATLAB script performs comparison between GM surface thickness of two populations of subjects that have already been processed using FreeSurfer anatomical processing pipeline(recon-all)

Edit

Add topics

4 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

Update README.md — Latest commit 14bf331 on Jun 26

BatchSurfTest.m	Just corrected some structure-same code	3 months ago
FreeSurfGroupTtest.sh	Just corrected some structure-same code	3 months ago
README.md	Update README.md	3 months ago
SurfTtest.m	Just corrected some structure-same code	3 months ago

README.md

FSGroupSurfTtest

This MATLAB script performs comparison between GM surface thicknesses of two populations of subjects that have already been processed using FreeSurfer anatomical processing pipeline(recon-all)

You need MATLAB and FreeSurfer to execute this code.

Sample usage(MATLAB script): SurfTtest(P1,P2,'P1vsP2','P1','P2','PN');

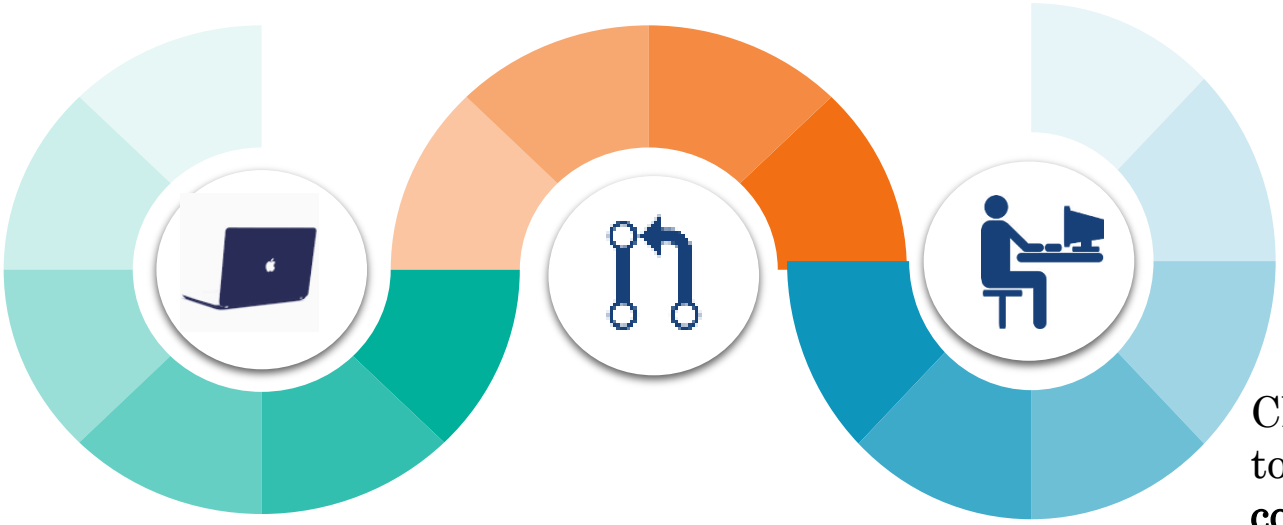
P1 and P2 are the names of the subjects in two groups being compared. These have to be MATLAB strings. Each of these subjects must have already been processed using FreeSurfer processing pipeline(recon-all or similar) and their surface stats must be present in $FREESURFER_DIR/subjects/Sub1/surf/ directory.

The directory 'P1vsP2' contains all the intermediate/FSGD etc files created.

US

Using source control–steps

Connect local code directory with Git repo
Commit code



Create **Github account**
Create new rep on Github
Install **TortoiseSVN**

Checkout from Git to **make working copy**
Work on it
Commit and add comments

"*Code complete*, A practical handbook of software construction" Steve McConnell

Some tips to write professional code

- 1: Don’t write code
  - I/O, sort, search, webIO, images…
  - Instead, reuse…

“*Code complete*, A practical handbook of software construction” Steve McConnell

Some tips to write professional code

- 2: Don’t do hard things, do easy things
  - Avoid tricky algorithms that you don’t understand
  - Create a new module for a new task
- 3: Re-factoring > Rewriting
- 4: Rewriting > Patching

“*Code complete*, A practical handbook of software construction” Steve McConnell