Three habits to bridge research code and sustainable software

January 21, 2016

Chris Gates (cgates@umich.edu)
UM Bioinformatic Core
DCMB Masters student
Agenda

• Background / Motivations
• Research code and sustainable software
• Habits:
  • Version control
  • Testing
  • Pairing
Background / motivations

• About me
  • 20 yrs in IT; 15 years in Software Engineering
  • 2007 Compendia (Oncomine)
  • 2012 UM Bioinformatics Core
    • What the core does
  • What I do
    • IT
    • Bioinformatics projects
    • Software engineering
Research code and sustainable software are often distinct

<table>
<thead>
<tr>
<th></th>
<th>Research code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optimized for</strong></td>
<td>Enabling discovery</td>
</tr>
<tr>
<td><strong>Interaction</strong></td>
<td>Informal; Interactive exploration</td>
</tr>
<tr>
<td><strong>Operational knowledge</strong></td>
<td>Implicit (authors)</td>
</tr>
<tr>
<td><strong>Target users</strong></td>
<td>Authors; Subject Experts</td>
</tr>
<tr>
<td><strong>Lifecycle</strong></td>
<td>Short; mostly development</td>
</tr>
</tbody>
</table>

Heroux (2009), Wilson (2014)
Operational profiles create distinct niches

Target “users”
(# users + maintainers + environments)

Lifecycle (# executions)

- Research code niche
- Sustainable software niche
Research code and sustainable software share many values

<table>
<thead>
<tr>
<th>Research code</th>
<th>Sustainable software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reproducibility</td>
<td>Robustness</td>
</tr>
<tr>
<td></td>
<td>Portability</td>
</tr>
<tr>
<td></td>
<td>Reusability</td>
</tr>
<tr>
<td>Early publication</td>
<td>Time to market</td>
</tr>
<tr>
<td>Correctness</td>
<td></td>
</tr>
<tr>
<td>Simplicity</td>
<td></td>
</tr>
</tbody>
</table>
I. Version control is a lab notebook for files
You use version control now

- floss
- stop cursing
- lose weight

- floss
- stop cursing
- lose weight
- exercise more
- cut out sugar

- floss 1x a day
- limit cursing in front on kids
- lose 10 pounds by April 15
- exercise 2x a week
- cut out desserts after lunch
But using a file system as version control is problematic

- Which is the most current file?
- What is the order of revisions?
  - What version did I use on December 23?
- Why was the file changed on Jan 1, 2016?
  - Who made that change?
Version control using Git

- floss
- stop cursing
- lose weight

- floss
- stop cursing
- lose weight
- exercise more
- cut out sugar

- floss 1x a day
- limit cursing in front on kids
- lose 10 pounds by April 15
- exercise 2x a week
- cut out desserts after lunch
Version control using Git

$ ls
resolutions.txt
$ git history
2015-12-01 cgates 02ac095 initial commit
2015-12-05 cgates 779a6dc added a few more
2015-12-21 cgates 8243fd0 made goals objective
2016-01-01 cgates a808ee1 made goals more realistic

• Which is the most current file? (resolutions.txt)
• What is the order of revisions? (as above)
  • What version did I use on December 23? (made goals objective)
• Why was the file changed on 1/1/2016? (“more realistic”)
  • Who made that change? (cgates)
Benefits of Git and GitHub

• **Git**
  - Provenance and history
  - Simpler/cleaner
  - Backup

• **GitHub** (Hosted version control)
  - Free for public projects
  - Better backup
  - Collaboration
    - Sharing
    - Publishing
    - Cooperative development

Chacon (Pro Git), Wilson (2014),
Version Control: Threats to adoption

• Big files
  • Don’t version control things you don’t edit by hand

• Privacy
  • Github/Bitbucket – cheap private accounts
  • Private hosting is easy for basic projects
II. Testing

- Code and fix (ad hoc testing)

- Traditional (waterfall) software development lifecycle

- Unit testing (Automated, iterative testing)

- Test-driven development (TDD)

Either is great

TDD Example: Roman Numerals

I ➞ 1
II ➞ 2
V ➞ 5
Testing influences your design

Hard to test

Classifier

Plotter

Easier to test

More modular

Beck (2003), Sandve (2013)
Benefits of automated unit testing

- Improves correctness during development
- Encourages re-use
- Passing tests quantify progress
- Reduces regressions over time
- Typically correlates with higher quality than “code and fix”

<table>
<thead>
<tr>
<th>BfxCore projects</th>
<th>Unit tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>AmpliconSoftClipper</td>
<td>71</td>
</tr>
<tr>
<td>Epee</td>
<td>717</td>
</tr>
<tr>
<td>Jacquard</td>
<td>537</td>
</tr>
<tr>
<td>Nephroseq</td>
<td>8315</td>
</tr>
<tr>
<td>Zither</td>
<td>53</td>
</tr>
</tbody>
</table>

Testing: Threats to adoption

- Stochastic algorithms harder (use/allow seeding)
- Big data slower (use small data)
- UI hard to test (separate data and presentation)
- Benefit smaller on simpler problems
- Startup cost
- Testing doesn’t guarantee correct behavior (thanks, Volkswagen!)
- Need a good problem model
III. Pair-programming

Two people, one keyboard

Economics of pairing

Parallel development (conventional)

- Ted (dev) | Project 1
- Amanda (dev) | Project 2

3 months * 2 dev = 6 dev months

If development were about typing, you would expect:

- Ted | Project 1
- Amanda | Project 2

6 months * 2 dev = 12 dev months

But in actuality, developing is more about problem solving:

- Ted | Project 1 | Project 2
- Amanda

3.3 months * 2 dev = 6.6 dev months
Benefits of pairing

Actual results

Ted
Amanda

• (120% effort)
• Shared understanding
• Homogenous code
• Early publication
• Simpler management

## Pairing at BfxCore

### Projects

<table>
<thead>
<tr>
<th>AmpliconSoftClipper</th>
<th>CRIDA</th>
<th>Epee</th>
<th>Jacquard</th>
<th>Nephroseq</th>
<th>Zither</th>
<th>(others)</th>
</tr>
</thead>
</table>

```python
def test_softclip_target_edgeInsert(self):
    util = cigar.CigarUtil(42, "3M" "1I4M" "2X")
    #444 444445
    #234 567890
    #ATAAACGTAC
    #MMMI
    # MMMM
    # XX
    #SSSSMMMMSS
    new_util = util.softclip_target(45,49)
    self.assertEquals("4S" "4M" "2S", new_util.cigar)
    self.assertEquals(45, new_util.reference_start)
```
Pair-programming: threats to adoption

- Logistics
- Mentorship
- Culture of individual ownership
- Furniture
### Habits can benefit both research code and sustainable software

<table>
<thead>
<tr>
<th>Value</th>
<th>Habit</th>
<th>Version control</th>
<th>Testing</th>
<th>Pairing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reproducibility</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Correctness</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Publication</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Simplicity</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

- Habits don’t make good decisions; they just make bad decisions more painful.
- Note that adoption of any habit (including good habit) reduces efficiency at the outset.
- Wilson (2014): Science is more than a body of knowledge – it’s a way of doing things that enables and encourages collaboration.
Thanks and questions

- Bioinformatics core
- Ana Grant
- Bob Boguski
- Divya Kriti
- Pete Ulintz
- Jessica Bene
- Kevin Meng
- Ross Patterson
References (1)

- UM BIOINF575
- UM EECS398
- Software carpentry: http://software-carpentry.org/
- Software Sustainability Institute: http://www.software.ac.uk/
References (2)