CIS 602-01: Computational Reproducibility

Data Sharing

Dr. David Koop
Assignment 1

• http://www.cis.umassd.edu/~dkoop/cis602/assignment1.html
• Must have a GitHub account (free)
• Turn in a link to your GitHub repo to myCourses
• Due Friday, October 7
• Uses shell scripts
  - Use Terminal.app on Mac OS X, the GitHub Desktop shell on Windows, and your favorite console on Linux
  - For Windows, you can get into bash by typing bash in PowerShell
• Please continue let me know if you have questions or find bugs!
• Fixing errors: git revert or git reset (--hard)
• Merge: use a merge tool or consider (--ours/their)
Git Flow

Feature branches

Develop branches

Hotfixes

Master

Feature for future release

Major feature for next release

Incorporate bugfix in develop

Bugfix from rel. branch may be continuously merged back into develop

Severe bug fixed for production: hotfix 0.2

Start of release branch for 1.0

Only bugfixes!

Tag 0.1

Tag 0.2

Tag 1.0

Time

[V. Driessen, CC BY-SA, 2010]
GitHub Flow

CREATE A BRANCH
Create a branch in your project where you can safely experiment and make changes.

OPEN A PULL REQUEST
Use a pull request to get feedback on your changes from people down the hall or ten time zones away.

MERGE AND DEPLOY
Merge your changes into your master branch and deploy your code.

ADD COMMITS

DISCUSS AND REVIEW

[GitHub Flow]
GitHub Study

• Interview Participants:
  - (heavy, light) x (hobbyist, work (non-SW org), work (SW org))

• Results:

<table>
<thead>
<tr>
<th>Visible Cues</th>
<th>Social Inferences</th>
<th>Representative Quote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recency and volume of activity</td>
<td>Interest and level of commitment</td>
<td>“this guy on Mongoid is just -- a machine, he just keeps cranking out code.” (P23)</td>
</tr>
<tr>
<td>Sequence of actions over time</td>
<td>Intention behind action</td>
<td>“Commits tell a story. Convey direction you are trying to go with the code … revealing what you want to do.” (P13)</td>
</tr>
<tr>
<td>Attention to artifacts and people</td>
<td>Importance to community</td>
<td>“The number of people watching a project or people interested in the project, obviously it’s a better project than versus something that has no one else interested in it.” (P17)</td>
</tr>
<tr>
<td>Detailed information about an action</td>
<td>Personal relevance and impact</td>
<td>“If there was something [in the feed] that would preclude a feature that I would want it would give me a chance to add input to it.” (P4)</td>
</tr>
</tbody>
</table>

[Dabbish et al., 2012]
Social Inferences

- Project management
- Learning from others
- Managing reputation and status
GitHub Actions and Reproducibility

- Do actions with respect to reproducibility signal anything?
- Is there a social model for reproducibility?
- What faults does GitHub have (e.g. gender studies)
- How do those faults impact reproducibility concerns?
Data about GitHub via its API

Events

CreateEvent
PushEvent
WatchEvent
ForkEvent
GitHub Event Information

```
{
    "type": "WatchEvent",
    "payload": {...},
    "public": true,
    "repo": {...},
    "created_at": "2012-05-28T12:42",
    "id": "1556481024",
    "actor": {"login": "Sarukhan"}
}
```
GitHub Entity Information

```
{
  "type": "User",
  "public_gists": 10,
  "login": "gousiosg",
  "followers": 64,
  "name": "Georgios Gousios",
  "public_repos": 20,
  "created_at": ...
  "id": 386172,
  "following": 16
}
```

[G. Gousios, 2016]
GHTorrent Stats

![Graph showing the number of events over time for various event types.](image-url)

[G. Gousios, 2016]
Factors affecting pull request acceptance

- requester_succ_rate
- prior_interaction_events
- perc_external_contribs
- workload
- pr_ratio
- team_size
- description_length
- prev_pullreqs
- prior_interaction_comments
- lang
- hotness
- commits_on_files_touched
- commits_to_hottest_file
- asserts_per_kloc
- test_cases_per_kloc
- num_commits_open
- followers
- test_lines_per_kloc
- entropy_diff
- files_modified_open
- src_files_open
- main_team_member
- src_churn_open
- files_changed_open
- new_entropy

[G. Gousios, 2016]
Can we handle the workload?

Which factors affect the time to process PRs?

Do we know the submitter?

How ready is our project for PRs?

What does the PR look like?

Factors affecting time to process PRs

[G. Gousios, 2016]
Diversity and GitHub

[B. Vasilescu et al., 2015]
Similarity attraction theory

People prefer working with others similar to them in terms of values, beliefs, and attitudes [Byrne]

Social identity and social categorization theory

People categorize themselves into specific groups. Members of own group are treated better than outsiders [Tajfel]

Due to greater perceived differences between groups than within groups, diversity can lead to confusion, stress, and conflict [Horwitz & Horwitz]

[B. Vasilescu et al., 2015]
Diversity Benefits

Driver of internal innovation and business growth [Forbes]

Diverse problem solvers outperform high ability problem solvers [Hong & Page]

Companies with diverse executive boards have higher earnings and returns on equity [McKinsey]

Multicultural social networks promote creativity [Harvard Business School]

[B. Vasilescu et al., 2015]
Gender diversity in GitHub teams

Gender diversity = mix women/men
simplifying assumption: gender is binary

The “hacker” culture is male-dominated and unfriendly to women [Turkle]

Women are <10% in OSS [Robles et al]

Reports of active discrimination and sexism towards women [Nafus]

[B. Vasilescu et al., 2015]
Study using GHTorrent data

Mining

- github
- social coding

Response

- Productivity (#commits/quarter)
- Turnover (fraction team new w.r.t. prev. quarter)

Independent

- Gender diversity (Blau index)
- Tenure diversity (coeff. variation)
  - project
  - overall coding

Controls

- Team size
- Time
- Project age
- Project activity

Sample 4K projects

[Study using GHTorrent data]

[B. Vasilescu et al., 2015]
Study Results

Productivity
(#commits/quarter)

Team size
Overall project activity
Forks
Project age

all team sizes
mid-size & large teams
Gender diversity
Tenure diversity

[19]

Study Results

Productivity
(#commits/quarter)

Team size
Overall project activity
Forks
Project age

all team sizes
mid-size & large teams
Gender diversity
Tenure diversity

[19]

Study Results

Productivity
(#commits/quarter)

Team size
Overall project activity
Forks
Project age

all team sizes
mid-size & large teams
Gender diversity
Tenure diversity

[19]

Study Results

Productivity
(#commits/quarter)

Team size
Overall project activity
Forks
Project age

all team sizes
mid-size & large teams
Gender diversity
Tenure diversity

[19]

Study Results

Productivity
(#commits/quarter)

Team size
Overall project activity
Forks
Project age

all team sizes
mid-size & large teams
Gender diversity
Tenure diversity

[19]

Study Results

Productivity
(#commits/quarter)

Team size
Overall project activity
Forks
Project age

all team sizes
mid-size & large teams
Gender diversity
Tenure diversity

[19]

Study Results

Productivity
(#commits/quarter)

Team size
Overall project activity
Forks
Project age

all team sizes
mid-size & large teams
Gender diversity
Tenure diversity

[19]

Study Results

Productivity
(#commits/quarter)

Team size
Overall project activity
Forks
Project age

all team sizes
mid-size & large teams
Gender diversity
Tenure diversity

[19]
Gitless Update

This is Git. It tracks collaborative work on projects through a beautiful distributed graph theory tree model.

Cool. How do we use it?

No idea. Just memorize these shell commands and type them to sync up. If you get errors, save your work elsewhere, delete the project, and download a fresh copy.

[xkcd, R. Munroe]
Purposes, Concepts, Misfits, and a Redesign of Git

Santiago Perez De Rosso  Daniel Jackson
Computer Science and Artificial Intelligence Lab
Massachusetts Institute of Technology
Cambridge, MA, USA
{sperezde, dnj}@csail.mit.edu

Abstract

Git is a widely used version control system that is powerful but complicated. Its complexity may not be an inevitable consequence of its power but rather evidence of flaws in its design. To explore this hypothesis, we analyzed the design of Git using a theory that identifies concepts, purposes, and misfits. Some well-known difficulties with Git are described, and explained as misfits in which underlying concepts fail to meet their intended purposes. Based on this analysis, we designed a reworking of Git (called Gitless) that attempts to remedy these flaws.

To correlate misfits with issues reported by users, we conducted a study of Stack Overflow questions. And to determine whether users experienced fewer complications using Gitless in place of Git, we conducted a small user study. Results suggest our approach can be profitable in identifying, analyzing, and fixing design problems.

Categories and Subject Descriptors  D.2.2 [Software Engineering]: Design Tools and Techniques; D.2.7 [Software Engineering]: Distribution, Maintenance and Enhancement—Version Control

Keywords  concepts; concept design; design; software design; usability; version control; Git.

1. Introduction

Experiment  This paper describes an experiment in software design. We took a popular software product that is both highly regarded for its functionality, flexibility and performance, and yet is also frequently criticized for its apparent complexity, especially by less expert users.

First, we did an analysis of the product, in which we applied some new design principles [16] in an attempt to identify problematic aspects of the design, suggesting respects in which the design might be improved. Since any such analysis is likely to be influenced by subjective factors (not least our own experiences using the product, and the particular contexts in which we used it), we corroborated the analysis by examining a large number of posts in a popular Q&A forum, to determine whether the issues we identified were in fact aligned with those that troubled other users.

Second, we reworked the design to repair the deficiencies identified by our analysis, and implemented the new design. To evaluate the redesign, we conducted a user study in which users with a range of levels of expertise were asked to complete a variety of tasks using the existing and new product. We measured the time they took, and obtained feedback on their subjective perceptions.

In some respects, this project has been a fool’s errand. We picked a product that was popular and widely used so as not to be investing effort in analyzing a strawman design, we thought that its popularity would mean that a larger audience would be interested in our experiment. In sharing our research with colleagues, however, we have discovered a significant polarization. Experts, who are deeply familiar with the product, have learned its many intricacies, developed complex, customized workflows, and regularly exploit its most elaborate features, are often defensive and resistant to the suggestion that the design has flaws. In contrast, less intensive users, who have given up on understanding the product, and rely on only a handful of memorized commands, are so frustrated by their experience that an analysis like ours seems to them belaboring the obvious.

Nevertheless, we hope that the reader will approach our analysis with an open mind. Although our analysis and experiment are far from perfect, we believe they contribute new ideas to an area that is important and much discussed by practitioners, but rarely studied by the research community.

Subject  Git, according to its webpage, is a free and open source distributed version control system that is easy to learn, has a tiny footprint, lightning fast performance, and features

• [De Rosso and Jackson, 2016]
• New Paper, same authors
• OOPSLA 2016
• More examples with categorizations
• Better comparisons
## Git Misfits and Stack Overflow Questions

<table>
<thead>
<tr>
<th>Misfit</th>
<th>Question</th>
<th>Upvotes</th>
<th>Views</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saving Changes</td>
<td>Q1 Using Git and Dropbox together effectively?</td>
<td>927</td>
<td>215523</td>
</tr>
<tr>
<td></td>
<td>Q2 Backup a Local Git Repository</td>
<td>122</td>
<td>78674</td>
</tr>
<tr>
<td></td>
<td>Q3 Fully backup a git repo?</td>
<td>54</td>
<td>37502</td>
</tr>
<tr>
<td></td>
<td>Q4 Is it possible to push a git stash to a remote repository?</td>
<td>105</td>
<td>30820</td>
</tr>
<tr>
<td></td>
<td>Q5 Git fatal: Reference has invalid format: refs/heads/master</td>
<td>90</td>
<td>25717</td>
</tr>
<tr>
<td></td>
<td>Q6 Is “git push --mirror” sufficient for backing up my repository?</td>
<td>34</td>
<td>18415</td>
</tr>
<tr>
<td></td>
<td>Q7 How to back up private branches in git</td>
<td>33</td>
<td>10580</td>
</tr>
<tr>
<td>Switching Branches</td>
<td>Q8 The following untracked working tree files would be overwritten by checkout</td>
<td>365</td>
<td>378331</td>
</tr>
<tr>
<td></td>
<td>Q9 git: Switch branch and ignore any changes without committing</td>
<td>148</td>
<td>129120</td>
</tr>
<tr>
<td></td>
<td>Q10 Why git keeps showing my changes when I switch branches (modified, added, deleted files) no matter if I run git add or not?</td>
<td>47</td>
<td>10524</td>
</tr>
<tr>
<td>Detached Head</td>
<td>Q11 Git: How can I reconcile detached HEAD with master/origin?</td>
<td>784</td>
<td>397694</td>
</tr>
<tr>
<td></td>
<td>Q12 Fix a Git detached head?</td>
<td>490</td>
<td>397985</td>
</tr>
<tr>
<td></td>
<td>Q13 Checkout GIT tag</td>
<td>125</td>
<td>98328</td>
</tr>
<tr>
<td>File Rename</td>
<td>Q14 git push says everything up-to-date even though I have local changes</td>
<td>113</td>
<td>79203</td>
</tr>
<tr>
<td></td>
<td>Q15 Why did my Git repo enter a detached HEAD state?</td>
<td>202</td>
<td>78856</td>
</tr>
<tr>
<td></td>
<td>Q16 Why did git set us on (no branch)?</td>
<td>65</td>
<td>41866</td>
</tr>
<tr>
<td></td>
<td>Q17 gitx How do I get my ‘Detached HEAD’ commits back into master</td>
<td>136</td>
<td>42794</td>
</tr>
<tr>
<td>File Tracking</td>
<td>Q18 Handling file renames in git</td>
<td>315</td>
<td>242864</td>
</tr>
<tr>
<td></td>
<td>Q19 Is it possible to move/rename files in git and maintain their history?</td>
<td>367</td>
<td>153701</td>
</tr>
<tr>
<td></td>
<td>Q20 Why might git log not show history for a moved file, and what can I do about it?</td>
<td>34</td>
<td>17099</td>
</tr>
<tr>
<td></td>
<td>Q21 How to REALLY show logs of renamed files with git?</td>
<td>60</td>
<td>12923</td>
</tr>
<tr>
<td>Untracking File</td>
<td>Q22 Why does git commit not save my changes?</td>
<td>177</td>
<td>142189</td>
</tr>
<tr>
<td></td>
<td>Q23 Git commit all files using single command</td>
<td>165</td>
<td>141815</td>
</tr>
<tr>
<td></td>
<td>Q24 Ignore files that have already been committed to a Git repository</td>
<td>1588</td>
<td>387112</td>
</tr>
<tr>
<td></td>
<td>Q25 Stop tracking and ignore changes to a file in Git</td>
<td>975</td>
<td>353136</td>
</tr>
<tr>
<td></td>
<td>Q26 Making git “forget” about a file that was tracked but is now in .gitignore</td>
<td>1458</td>
<td>286435</td>
</tr>
<tr>
<td></td>
<td>Q27 git ignore files only locally</td>
<td>562</td>
<td>120700</td>
</tr>
<tr>
<td></td>
<td>Q28 Untrack files from git</td>
<td>218</td>
<td>140663</td>
</tr>
<tr>
<td></td>
<td>Q29 Git: How to remove file from index without deleting files from any repository</td>
<td>110</td>
<td>61498</td>
</tr>
<tr>
<td></td>
<td>Q30 Ignore modified (but not committed) files in git?</td>
<td>135</td>
<td>38293</td>
</tr>
<tr>
<td></td>
<td>Q31 Ignoring an already checked-in directory’s contents?</td>
<td>169</td>
<td>49692</td>
</tr>
<tr>
<td></td>
<td>Q32 Apply .gitignore rules to an existing repository [duplicate]</td>
<td>40</td>
<td>28286</td>
</tr>
<tr>
<td></td>
<td>Q33 undo git update-index --assume-unchanged &lt;file&gt;</td>
<td>165</td>
<td>37262</td>
</tr>
</tbody>
</table>

[De Rosso and Jackson, 2016]
Task Evaluation

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Add readme file</td>
<td>Create a new file (readme), track it, make another modification to it, and create a commit that includes all changes made to the file</td>
</tr>
<tr>
<td>2. Let users input weight in kilos</td>
<td>Create a new branch feat/kilos, switch to it, make a change and commit. We then ask them to make another change that is left uncommitted</td>
</tr>
<tr>
<td>3. Let users input height in meters</td>
<td>Create a new branch feat/meters, switch to it and make a change. The participant then needs to switch to master to fix a bug</td>
</tr>
<tr>
<td>4. Wrap with features</td>
<td>Go back to working on the kilos feature, which involves switching to feat/kilos branch and bringing back uncommitted changes</td>
</tr>
<tr>
<td>5. Fixing conflicts</td>
<td>Switch to another branch in the middle of conflicts</td>
</tr>
<tr>
<td>6. Code cleanup</td>
<td>Undo an unpushed commit (as if it never existed before)</td>
</tr>
</tbody>
</table>

Figure 1: Task completion time (minutes)

(a) Task 1  (b) Task 2  (c) Task 3  (d) Task 4  (e) Task 5  (f) Task 6

[De Rosso and Jackson, 2016]
Post-Study Survey Results

(a) Satisfaction
(b) Efficiency
(c) Difficulty
(d) Frustration
(e) Confusion
(f) Git versus Gitless

Post-study and post-study questionnaire results (1=strongly disagree, 4=neutral, 7=strongly agree), with standard errors. I enjoyed using Gitless. I found Gitless to be easier to learn than Git. I would continue using Gitless if I could.

Note that while results suggest that our redesign efforts were fruitful (especially for novices, without a notable negative impact on experts) this doesn't mean Gitless is a “better” VCS than Git. Our study focused only on misfits and did so in a controlled environment. A full evaluation of a VCS would require testing it in the context of large projects with complex requirements. Yet our results provide some empirical evidence that suggests our approach can be profitable in addressing design-related usability problems.

8.3 Threats to Validity

Internal
In addition to the conceptual model, the type (e.g., command language, direct manipulation), and quality of the user interface affects usability. This is not a major factor in our study, since Gitless has a command line interface that follows the same Unix conventions as Git; the only differences are in the command names (and of course their semantics).

External
The user study was conducted on only 11 people that are, or have previously been, affiliated with computer science at MIT and may not generalize to Git users in general. To mitigate this factor Gitless is available online for free, and anyone can download and try the tool. Our findings may

[De Rosso and Jackson, 2016]
The Conundrum of Sharing Research Data

C. L. Borgman
Data Sharing

• What is data sharing?
  - "...the release of research data for use by others"

• What is data?
  - "broadly inclusive"
  - digital literature (e.g. games), data and databases requiring computers and software (e.g. genomic sequencing, observational data (remote sensing), and generated or compiled information (by humans or machines)
  - physical and life sciences: most gathered/produced by researchers
  - social sciences: gather/produce, also obtain from public records
  - humanities: records from human culture (archives)

• dataset: grouping, content, relatedness, purpose
Categories of Data

• Observational: e.g. weather, may go across time and location
• Computational: data from computer model and simulation
• Experimental: lab or field experiment (may be replicated)
• Records: e.g. government records

• (via National Science Board)
The method is slow and too insensitive to distinguish between human and animal sources of bacteria. The more sophisticated method is quantitative polymerase chain reaction (qPCR), adapted from medical applications, which requires greater expertise and is much more expensive. This method is faster and more sensitive, but results will vary between laboratories due to choices of local protocols, filter material, machine type and model, and handling methods. Protocols and results are shared between partner laboratories seeking to perfect the method, but little other than the methods of data collection, protocols, and final curves might be reported in the journal articles. Biological samples are fragile; they degrade quickly or are destroyed in the analysis process.

At the other end of the specificity dimension are observatories, which are institutions for the observation and interpretation of natural phenomena. Examples include NEON and LTER in ecology (National Ecological Observatory Network, 2010; U.S. Long Term Ecological Research Network, 2010; Porter, 2010), GEON in the earth sciences (GEON, 2011; Ribes & Bowker, 2008), and synoptic sky surveys in astronomy (Panoramic Survey Telescope & Rapid Response System, 2009; Large Synoptic Sky Telescope, 2010; Sloan Digital Sky Survey, 2010). Observatories attempt to provide a comprehensive view of some whole entity or system, such as the earth or sky. Global climate modeling, for example, depends upon consistent data collection of climate phenomena around the world at agreed upon times, locations, and variables (Edwards, 2010).

The value of observatories lies in systematically capturing the same set of observations over long periods of time. Astronomical observatories are massive investments, intended to serve a large community. Investigators and others can mine the data to ask their own questions or to identify bases for comparison with data from other sources. Studies of the role of dust emission in star formation make use of observatory data. In this star dust scenario, a team of astrophysics researchers queries several data collections that hold observations at different wavelengths, extracting many years of observations taken in a specific star-forming region of interest. They apply several new methods of data analysis to model physical processes in star formation. By combining data from multiple observatories, they produce empirical results that enable them to propose a new theory. Typically the combined dataset is released when they publish the journal article describing their results.

**Scope of Data Collection.** The second dimension of Figure 1 is the scope of data collection. At one pole are exploratory observatories, aimed at describing phenomena, and at the other pole are theoretical observatories, aimed at modeling systems. The goal of research is on the vertical axis, with empirical research at the bottom and theoretical research at the top. The specific purpose of the research is on the horizontal axis, with descriptive research on the left and model-building research on the right. The scenarios shown in the graph are:

- **BQ:** Beach Quality
- **SD:** Star Dust
- **OS:** Online Survey
- **AR:** Archival Records

**Scenarios:**

BQ: Beach Quality
SD: Star Dust
OS: Online Survey
AR: Archival Records
Goal of Research

• "Observations of the physical universe occur at a unique place and time and can never be reconstructed."
Approaches in Handling Data

Scenarios:
- BQ: Beach Quality
- SD: Star Dust
- OS: Online Survey
- AR: Archival Records

Graph: Three axes representing:
- People Involved: Collaborative Teams vs. Individual Investigator
- Labor to Collect Data: By Hand vs. By Machine
- Labor to Process Data: Not explicitly shown in the diagram

The graph illustrates the distribution of approaches across different scenarios.
Labor to Process Data

- Machine-collected versus human
  - Citizen science
- "Generally speaking, the more handcrafted the data collection and the more labor-intensive the postprocessing for interpretation, the less likely that researchers will share their data."
Rationales for Sharing Data

The model proposed here is intended to provoke discussion about the rationales for sharing research data. It positions the various arguments and beneficiaries along two axes:

- **Arguments for Sharing**
  - **Research-driven**
  - **Public-driven**

- **Beneficiaries of Sharing**
  - **Data Producers**
  - **Data Users**

**Rationales**
- **R1**: Reproduce/verify
- **R2**: Serve public interest
- **R3**: Ask new questions
- **R4**: Advance research

The model shows how these rationales may conflict with the incentives of data producers and users, and how they may or may not align with the motivations of funding agencies and journals. The model is not exhaustive; it is offered as a useful framework for examining the complex interactions of players, policies, and practices in data sharing. The example of reproducibility or replication of research is viewed as the gold standard for science (Jasny, Chin, Chong, & Vignieri, 2011), yet it is the most problematic rationale of the four.

**Arguments for Sharing**

- **R1**: Reproduce or verify research
- **R2**: Serve public interest
- **R3**: Ask new questions
- **R4**: Advance research

**Benefits of Sharing**

- **Data Producers**
- **Data Users**

Neither dimension is absolute; the poles represent relative positions of people or situations. For example, a researcher or policy maker may make one argument on behalf of the producers of data and another on behalf of the users. Similarly, an argument made in the name of scholarly purposes may also serve the public good. These arguments and rationales in favor of sharing research data may conflict with the incentives of those whose work produces the data. Accordingly, discussion of the four rationales focuses most heavily on the concerns of data producers and on their abilities, motivations, and incentives to share their data.
Reproducibility

• "…it is the most problematic rationale for sharing research data"!
• Data is not enough
• Cannot reduce research to "mechanistic procedures"
• [Depends more on interpretation than data]
Data Sharing

• Difficult
• Need to see use?
• Data may rely on software
• Streaming data?
Next Time

• Two papers by Vines et al. that study data availability in biology papers

• Reading Response
  - Why is availability an issue?
  - What factors contribute to availability?
  - What solutions would help improve access to data?
  - How does data fit in with reproducibility?