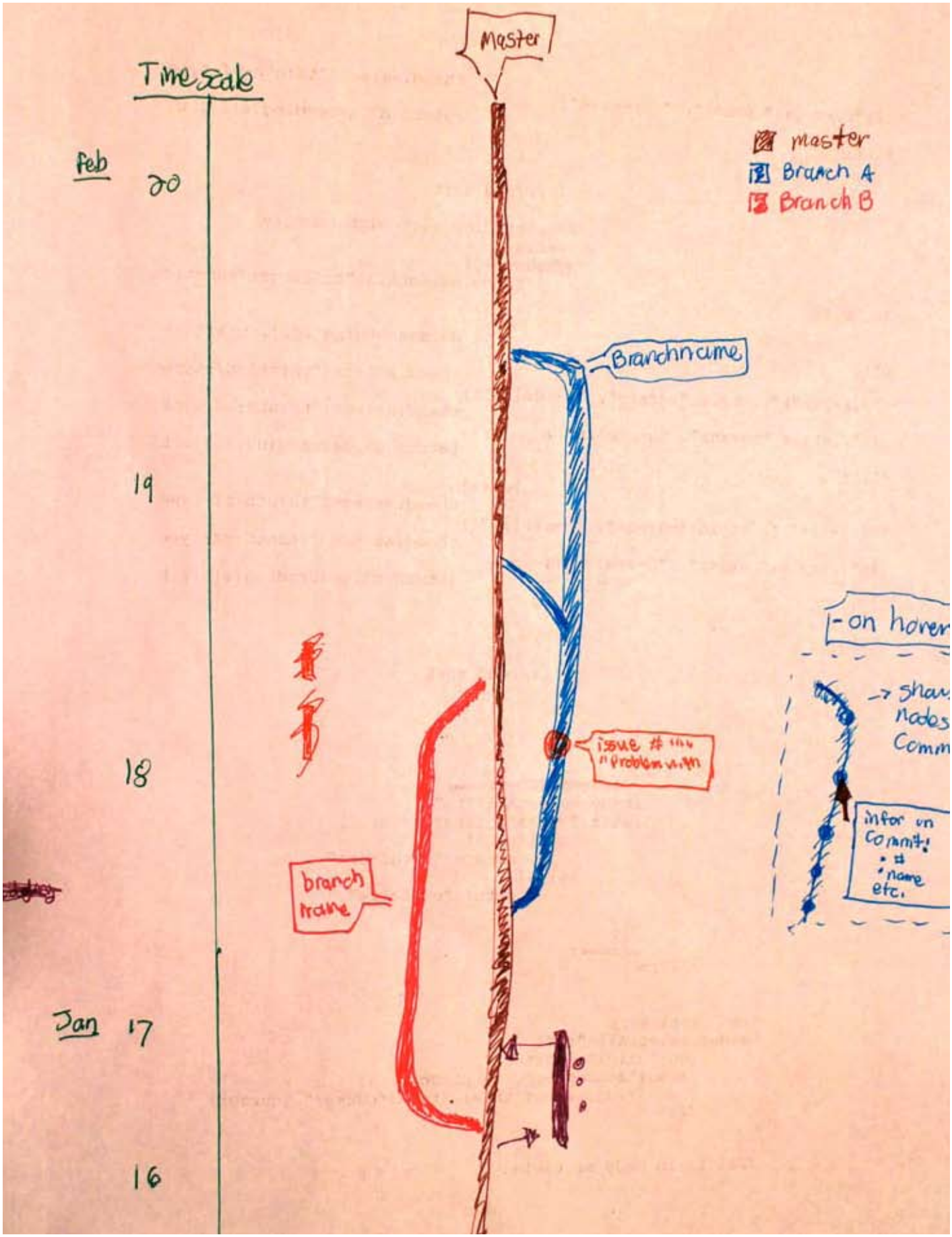
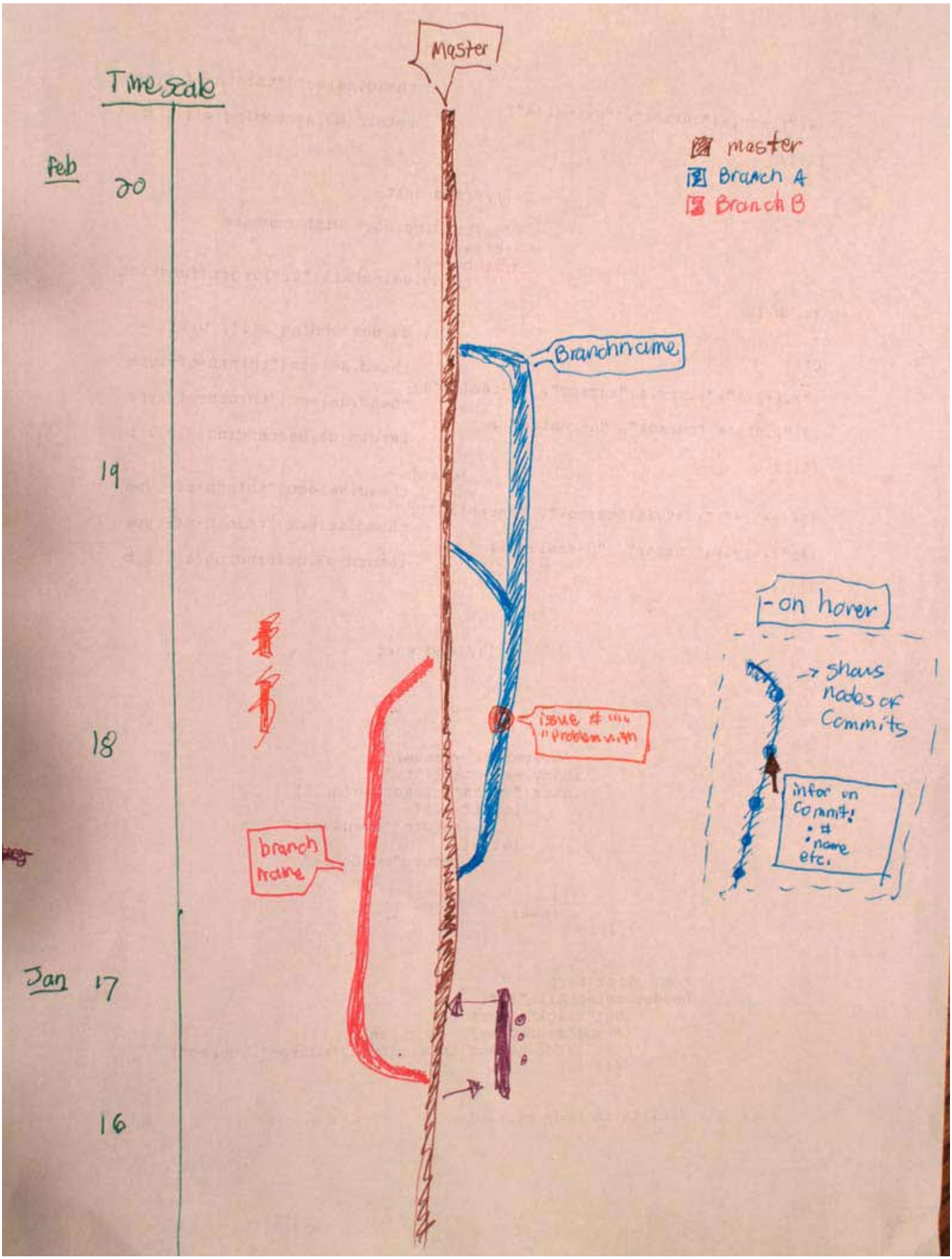
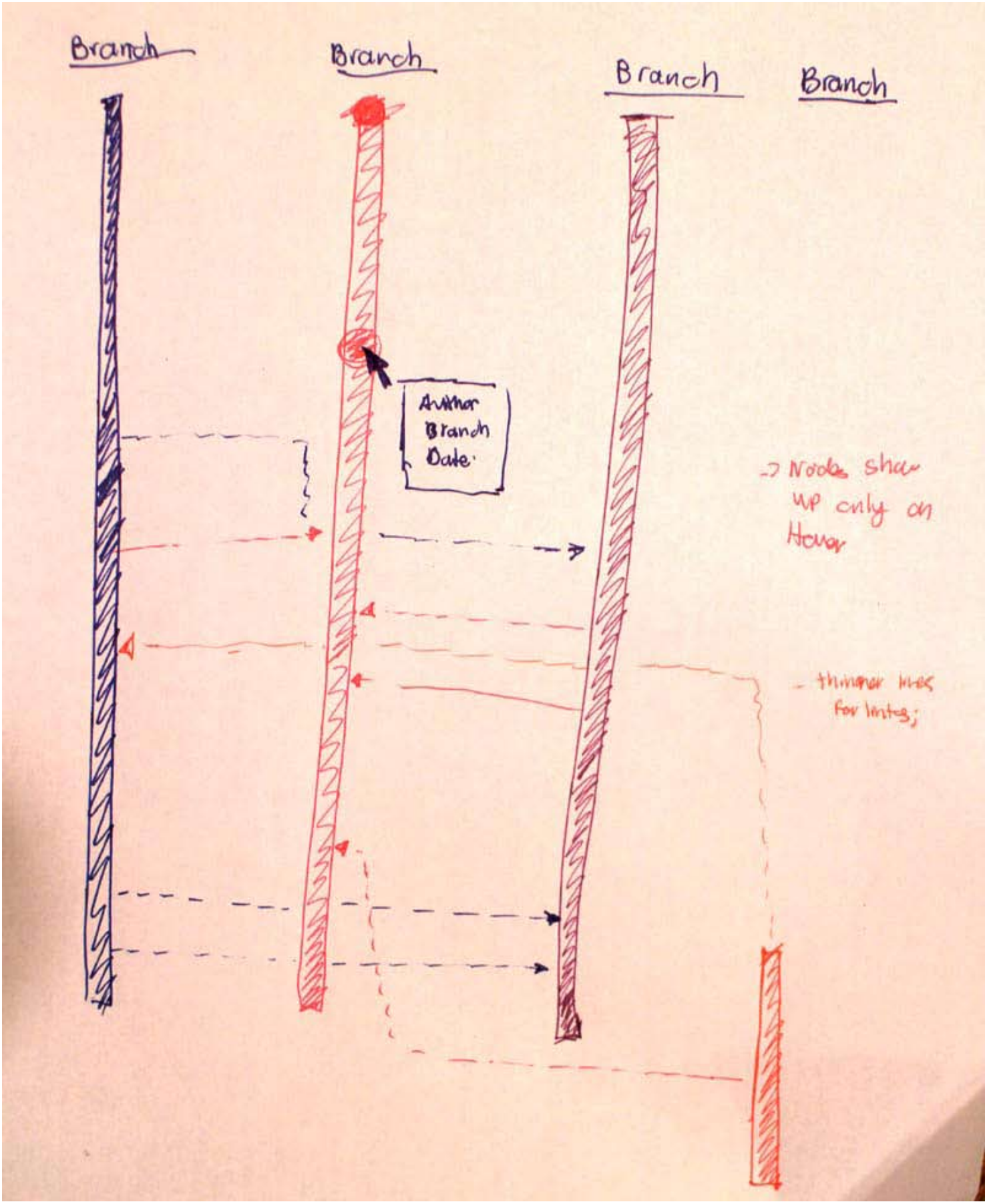
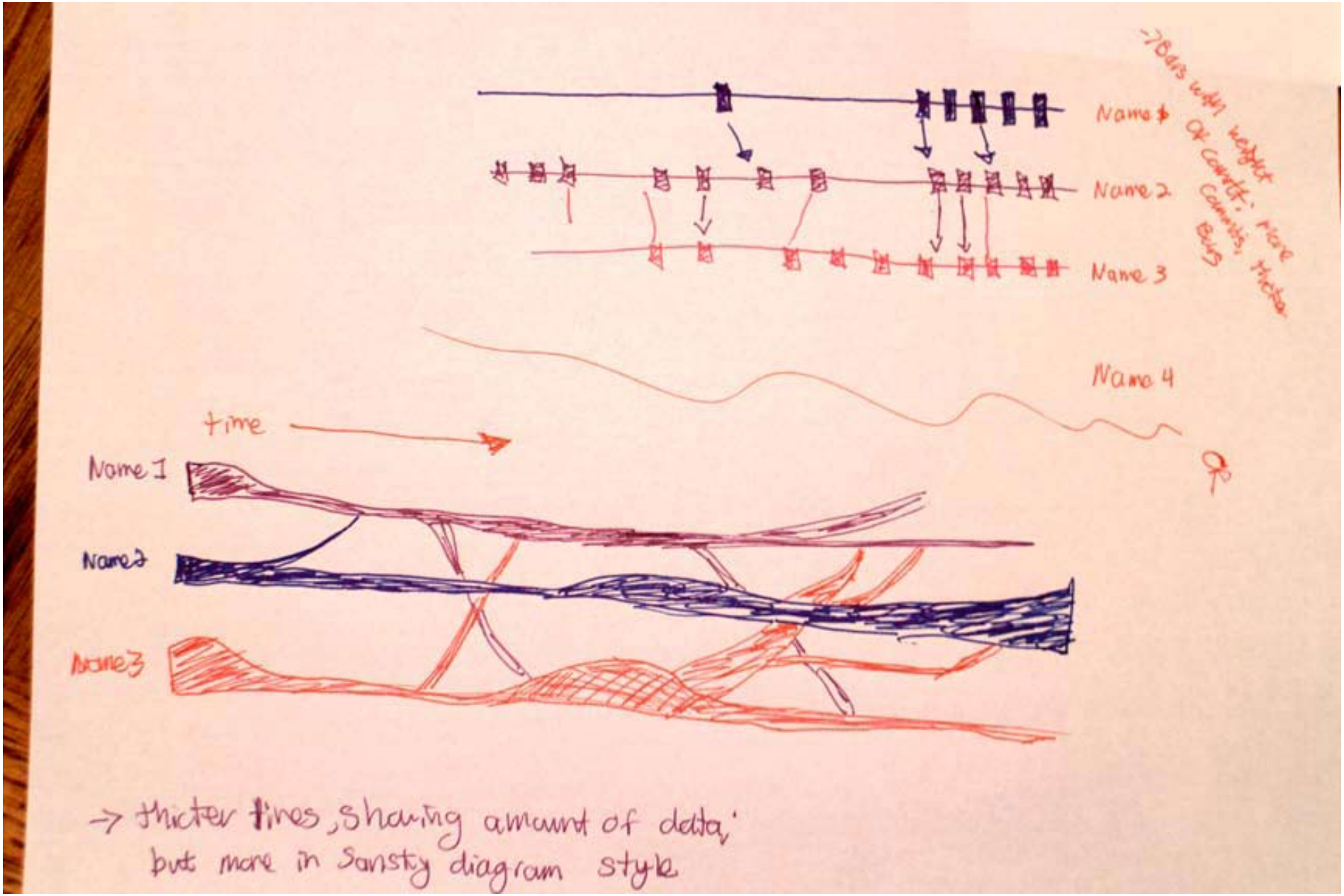

Timescale
Feb
20
19
18
Jan
17
16
master
Branch A
Branch B
master
Branchname
issue # 114
"Problem with
branch
name
-on hover
-> shar
nodes
Comm
infor on
Commit:
• #
• name
etc.


Timescale
Feb
20
19
18
Jan
17
16
Master
master
Branch A
Branch B
Branchname
branch name
issue # 1114
"problem with
-on hover
-> shows nodes or commits
infor on commit:
• #
• name
etc.


Branch
Branch
Branch
Branch
Author
Branch
Date
-> Nodes show up only on Hover
- thinner lines for lanes;


Name 1
Name 2
Name 3
Name 4
time
Name 1
Name 2
Name 3
OR
-> thicker lines, showing amount of data,
but more in Sankey diagram style

Design Studio & questions from homework part 3

Part 1. Analysis

The D3 page has a lot more contributions compared to the other GitHub graphs and as a consequence has a lot
of branches and commits for many users. THere are aslo a lot of users who aren't very active, or not very active recently, leaving large blank spots on the pages that is awkward to scroll through. Both D3 and Caleydo
have rather long time lines, CS171's is shorter with more frequent commits, making it seem more condensed.
D3's page has large x and y dimensions, Caleydo is more stretched across the x, with not as much y.
CS171 has less in both dimensions.

Attributes: project contributors, commits, in order of when committed, merges and branches

Other relevant attributes might be: Commit size, additions, deletions, current issues,
most active recent areas, active contributors, total daily commits

Filters that might be relevant: those who made commits, most active contributors,

Roles:

Caleydo: used for seeing what other branches are being developed and by who? Keeping track for merges?
D3: Maybe more specific to contributors wanting to see activities on the repository, or what commits they may not personal be current on. Also valuable for the project manager keeping track of project status, but because its so large unlike Caleydo, it is harder to get an overview of the development. With the D3 graph it seems there is more of a need to already know what is going on.
CS171: Most useful, for the class group, especially TA's, instructors, and graders. Could be useful, keeping track of updates and changes. With this graph I think it's less about developing features and more about managing information and data. Roles specific to this graph: students, graders, instructors, TF

Roles in all graphs: Manager, Viewers, Contributors, Interested Contributors

Task: Overview, status checks on: commits, branches, merges, contributors, checking on what have to do, who's working on what, perhaps checking for duplicate issues, amount of activity.

Focus for Project?
Role: Contributors
Tasks: Status check on activity on repository, viewing current activity with other users, viewing current issues, viewing commits, branches, merges,

Part 3:
In our online group discussion, we didn't pick one design or sketch, it wasn't really the most practical given the online situation, but we did have common components that

we agreed were important. They are:
1. Being able to see an overview of the repository
2. Showing more about the amount done with each commit, for example the additions and deletions.
3. Being able to filter out inactive users, to focus on important activity.

Design Sketch Paragraph: (From part 3 of homework assignment

In addressing the problem of clutter with larger data sets, my design has a vertical orientation instead of a horizontial one. This is easier to scroll through for a viewer and allows more of the data to be shown at one time. Ideally the view will expand to fit the data set. Also to eliminate some of the clutter in the graph, nodes or commits will only show up when the viewer hovers over the branch.

My design focuses on the big picture first and then has more details and information available upon exploration. I wanted to show the connections while still maintaining readability. Color and lines will be used to represent the connections themselves. Lines will show the branches, and vary in weight based on how many commits are made on each branch. It might by possible to create an effect similar to a Sankey diagram by doing this. Other lines will show the connections to other branches. These will be differentiated from the branch lines by weight. Shapes will be used to show the commits themselves, either circles or squares.

Ideally, my design would have multiple sorting features, allowing the viewer to see the data in multiple views and thus adjust to the varying users and their different work methods on GitHub.