Design 4

Bo-Shan Go	10780297
Ruben Gerritse	10760326
Thirza Dado	10492682
Mike La Grouw	10626700
Alexander de Groot	10078797

Part 1 Analysis

Think about how these networks are different. Analyze the "dimensions" of these networks. What are the relevant attributes (e.g., commits, users, branches, commit size, etc.) of these representations? What other attributes could be relevant in this graph? Write a list of all the attributes your visualization could show.
1: multiple dimensions (multiple branches). One branch is work in progress, and when finished it is added to the master branch.

2: only one master branch, and you don't know who changed what. One website; you do not want parallel processes (people that work at different parts of a project, and you do not want different versions of the same website).

Are there different roles, i.e., different types of users who might want to achieve different things? Write a list of user roles.
1: owner (Caleydo), admin (Jenkins), bug fixer (Samuel), technical working bee (Christian), designer bee (Thomas).
2: admin (Jenkins) and unknown working bees.

Think about which tasks a user of your visualization might want to achieve. Write down a list of tasks.
Overview: information about commits per person in time. Relationship between the commits. Clear distinction between different branches/projects by color.

Identify one role that you want to design your visualization for. Prioritize your task and attribute lists based on this role's needs.
Group-project: write an algorithm to solve a problem. Visualization of github network: one master branch with final version of the algorithm. Branches in different colors so that different members can experiment work on each other's versions, and commit to master branch when the experiments turn out good. Very alike visualization 1.

Part 2 sketching

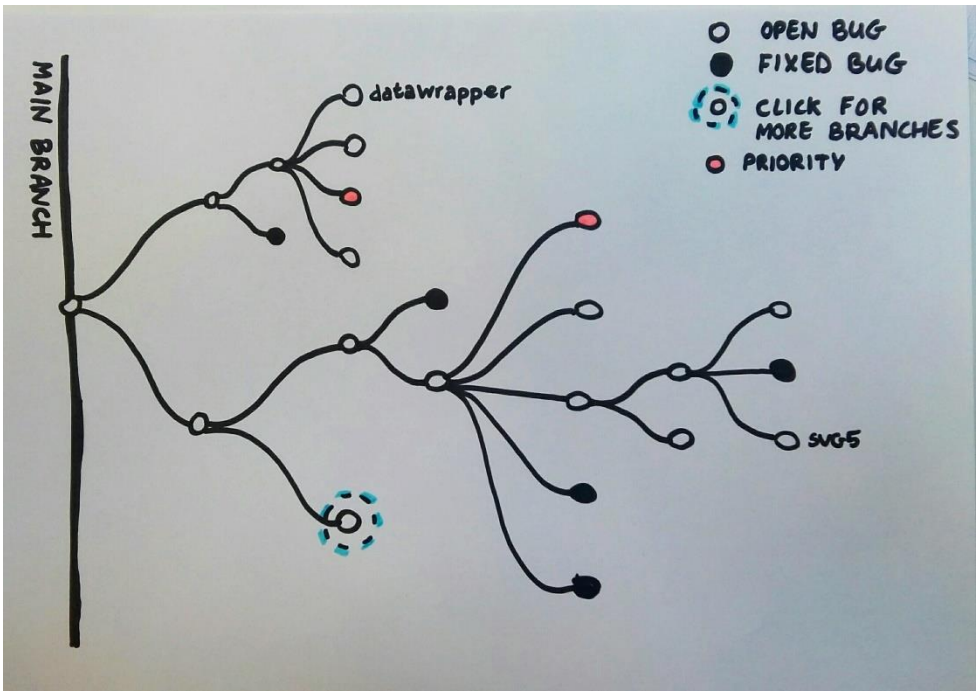

Figure 1 Bug fixer

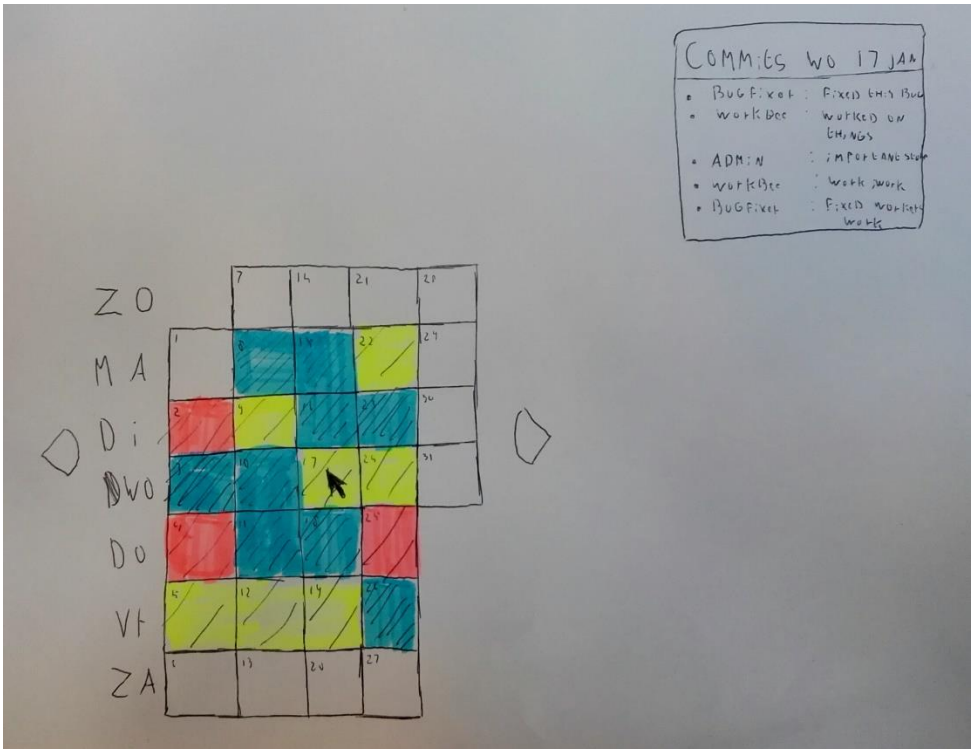

Figure 2 Calendar view

Decide on which visual variable to use for which attributes of the visualizations. Consider the strengths and weaknesses of visual variables that were mentioned in Carpendale's article Reading 2 (also briefly discussed in this week's lecture: Process). Use the strongest visual variable for the most important attributes of the data.
Figure 1: bug fixer. Capture order of changes in the visual representation. Flow in space to represent flow in order/time: the more to the right, the later in time.
Figure 2: activity in master branch. Calendar view. Color as strongest variable: blue as more changes in time whereas yellow represents less changes in time (represented by space; calendar view).

Do you think it is necessary to represent every single commit as a separate node? Could you think of ways to aggregate this?
Tree structure: every node is captured. However, multiple nodes for multiple file types (HTML, CSS, Javascript). Within each 'filetype-node' multiple members who commit a change are represented by a node with their own color. And every commits position in the visualization corresponds with the time of the commit (later is more to the right).

Do you think that every contributor needs a "row", as on the default network view on github? Could you think of a smarter way to summarize those?
No, because members should be able to work on different branches, each represented by a row in space.

Is a node-link diagram the appropriate representation? Or should you consider alternative graph representations?
Yes, it is very clear.

Part 3

Tree structure: overview perhaps not ideal. However, a good way to unmask common bugs, how to fix them, and the member that fixes it.

Calendar view: total overview about how the working goes. However, the information is very dense, and therefore does not provide detailed information about commits in separate branches by different members.