Scenario

Maintenance of software has been found to be very costly, impacting up to 80% of the overall cost and effort [1]. Fixing a bug or refactoring a piece of code are two of the maintenance activities developers often have to do. The most amount of time developers spend to do a fix or refactoring is understanding the piece of the software where change needs to be done. When the developer figured out the class where the change needs to be done, following information about the system can help a developer indicate how complex the task will be, therefore how long it will take to do the change:

* Dependencies of the class to other classes within the software system
* Strength of the dependencies
* Cyclometic complexity of each class
* Bigness of each class (TLOC)

List of Aesthetics Criteria & Our application of these criteria

Firstly, as the study of Purchase [2] shows, following aesthetic criteria have significant effect on understandability of graphics on humans, we decided to apply them to our visualization. Each aesthetic is given an acronym and the meaning is shown in figure 1.

1. Minimizing bends (b)
2. Minimizing edge crossings (c)
3. Maximizing minimum angles (m)
4. Maximizing orthogonality (o)
5. Symmetry (s)

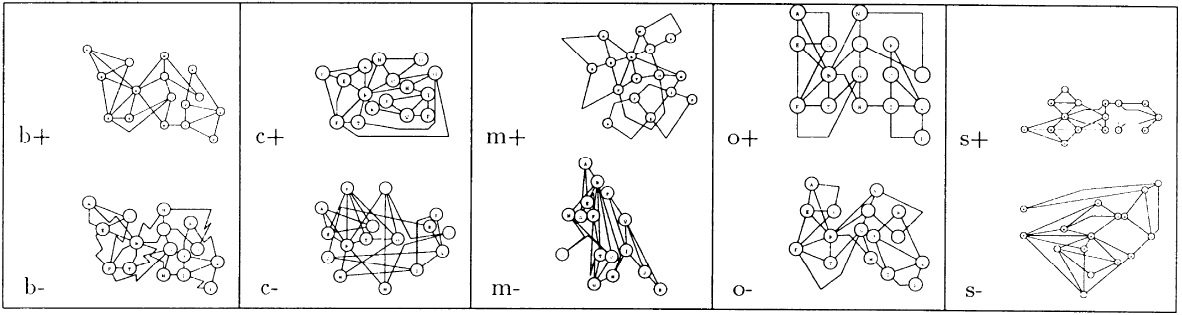


Figure 1. Explanation of the aesthetic criteria 1 to 5

Next to these, other aesthetic criteria we applied to our visualization are as following:

1. Maximizing similarity of shapes
2. Maximizing similarity of distance between border shapes and the page
3. Maximizing similarity of distance between shapes

List of Usability Criteria & Our application of these criteria

Below is a list of important usability criteria and per criteria list of decisions we made to satisfy the criteria in our solution.

* Performance
  + Extract data from the software system prior to visualization
  + Extract dependency tree per class to minimize processing time on the page
  + Use json format to store extracted data
  + Use JavaScript to find the correct file with data and visualization
* Responsiveness
  + Hover & tooltip
  + Hover & highlight dependencies on that line
  + Click on class box & redirect to the class on eclipse
* Easy & Quick Identifiability
  + Usage of opposite colors for CC
* Error prevention
  + Search combo box
* Understandability
  + Legend

Argumentation

References

[1] G. Alkhatib. The maintenance problem of application software: an empirical analysis. *Journal of Software Maintenance, 4(2):83–104*, 1992.

[2] H.C. Purchase, Effective information visualisation: a study of graph drawing aesthetics and algorithms. *The Department of Computer Science and Electrical Engineering, The University of Queensland, 2000*