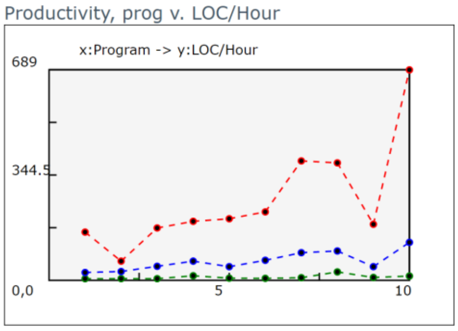
Analysis of Defects and Yield.

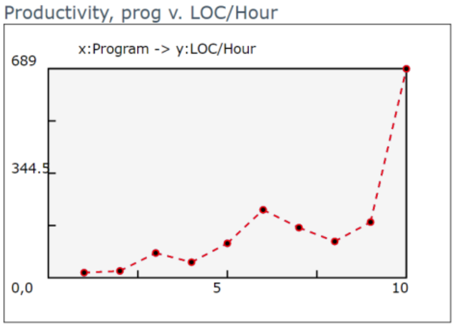
Defects and Yield is a measurement for productivity of developers. Here is the data collect from the ten program I did in this semester compare to the whole class.

- Productivity, prog v. LOC/Hour



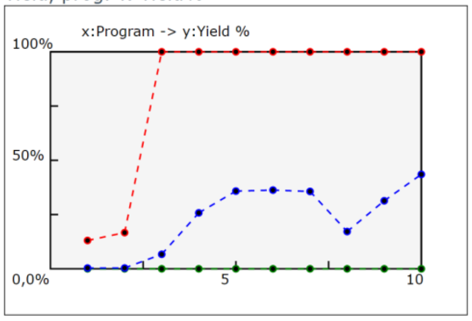
Class

- Productivity, prog v. LOC/Hour



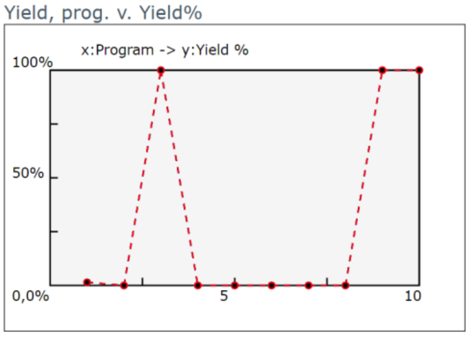
Myself

- Yield, prog. v. Yield%



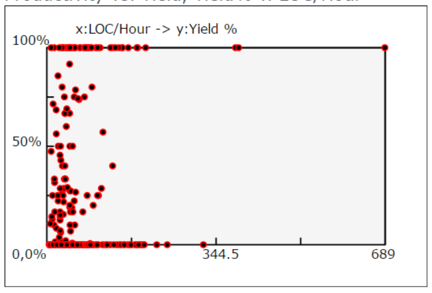
Class

- Yield, prog. v. Yield%



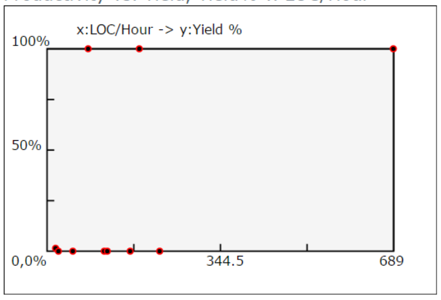
Myself

- Productivity vs. Yield, Yield% v. LOC/Hour



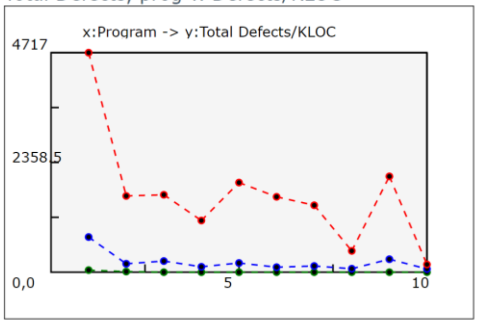
Class

- Productivity vs. Yield, Yield% v. LOC/Hour



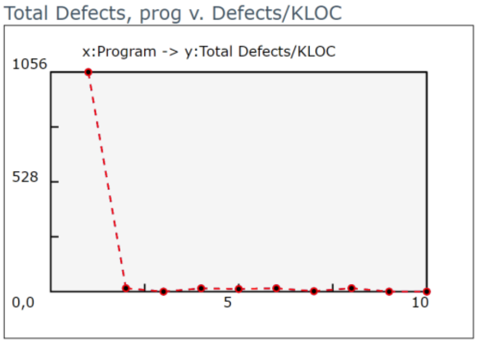
Myself

- Total Defects, prog v. Defects/KLOC



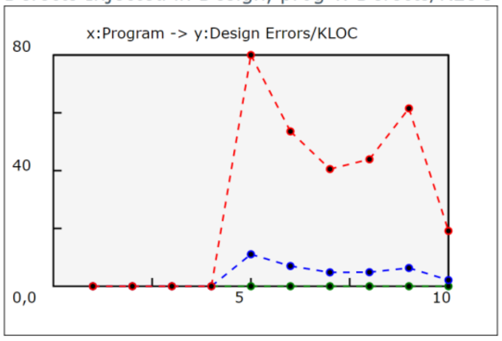
Class

- Total Defects, prog v. Defects/KLOC



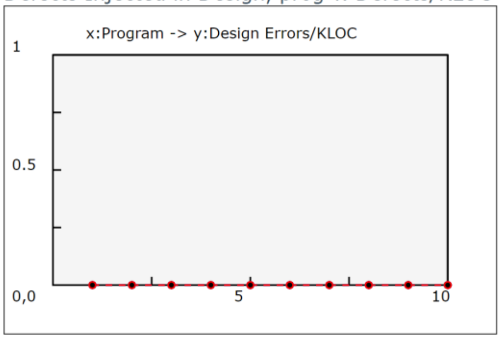
Myself

▪ Defects Injected in Design, prog v. Defects/KLOC



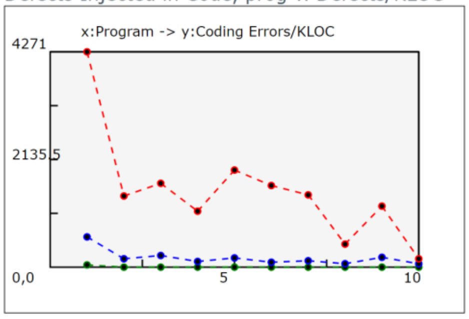
Class

▪ Defects Injected in Design, prog v. Defects/KLOC



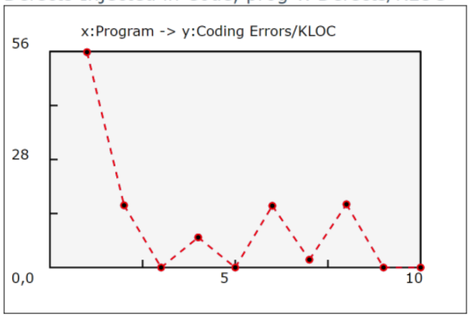
Myself

▪ Defects Injected in Code, prog v. Defects/KLOC



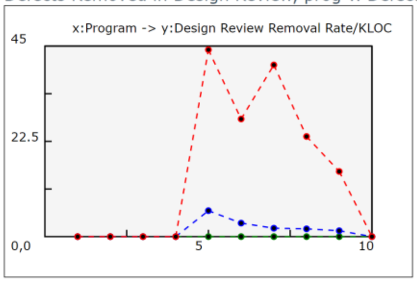
Class

▪ Defects Injected in Code, prog v. Defects/KLOC



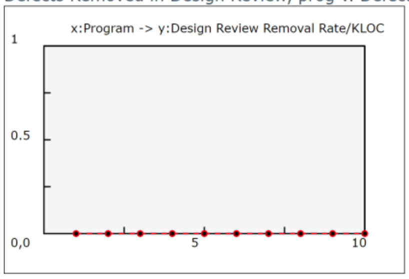
Myself

▪ Defects Removed in Design Review, prog v. Defects/KLOC



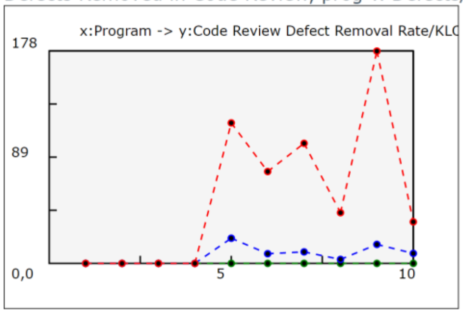
Class

▪ Defects Removed in Design Review, prog v. Defects/KLOC



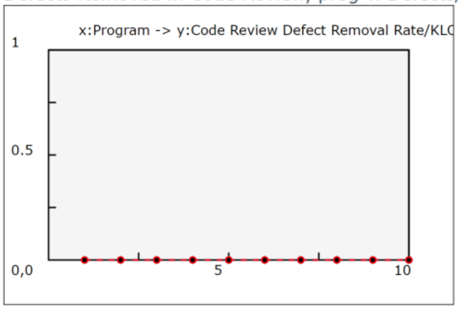
Myself

▪ Defects Removed in Code Review, prog v. Defects/KLOC



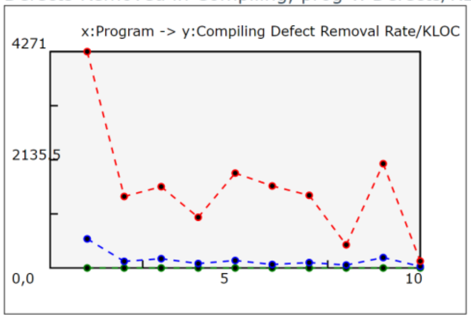
Class

▪ Defects Removed in Code Review, prog v. Defects/KLOC



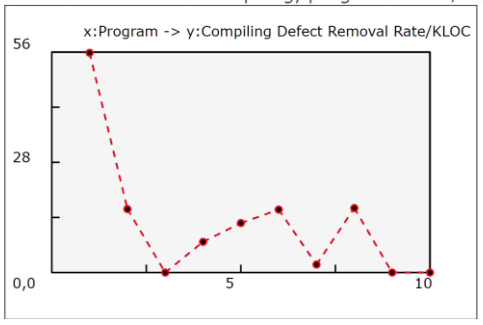
Myself

▪ Defects Removed in Compiling, prog v. Defects/KLOC



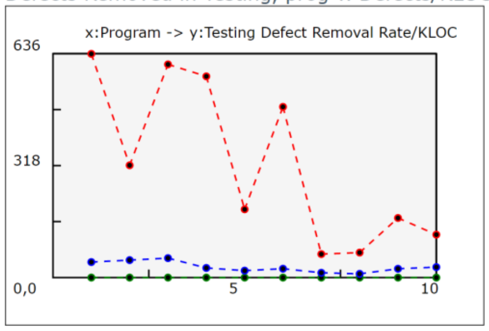
Class

▪ Defects Removed in Compiling, prog v. Defects/KLOC



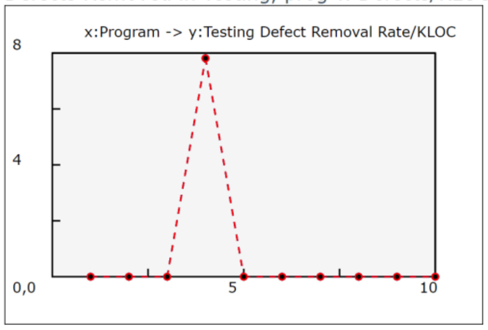
Myself

▪ Defects Removed in Testing, prog v. Defects/KLOC



Class

▪ Defects Removed in Testing, prog v. Defects/KLOC



Myself

From the diagram, I can draw some conclusion through the analysis.

1. The defect that I injected is all in coding. And after saw the defect log I found out that the main defect type is I forgot to add or delete spaces, tabs, comments.
2. The defect I have all inject in the coding part and the trend is going low through time. The first program have the most defect, and after that there are much few defect and at last program there is no defects.
3. The total trend of the defect also going low through time. There is much fewer defect after the first program. In the first program I have more than 1000 defect but after the first program, when I take the process carefully I only have less than 10 or even don't have any defect in each of the program.
4. The trend of the defect is going low through time. There is only 8 defects are remove from testing at program 04. Rest of the defect are remove from compiling. As we can see from the diagram, the first program have the most defect and at program10 there is no defect.
5. As we can see from the productivity vs yield, the yield is almost 100% when it comes to the programs that have the algorithm that I am familiar to. In the same time, if the algorithm I am not familiar with the yield is almost 0%
6. The trend of yield vs the A/FR from 5 to 10 is increasing. At program 9 and 10 there is a big increase because the algorithm is familiar to me.