Review 8

2/13/12

CS6V81.502

Seungtack Baek

**Detecting Software Modularity Violations**

In this paper, Wong et al. tried to detect modularity violation, using a tool named CLIO. CLIO first determines how elements should change together through structural coupling. CLIO can find structural coupling via Baldwin and Clark’s design rule and design structure matrix. Then CLIO finds what has actually been changed together by linking revision history. Lastly, this tool matches the results from former two steps and determines discrepancy between those, thus modularity violation.

With this tool, they analyzed several builds of Eclipse JDT and Hadoop. After analysis, they were able to detect modularity violation. As a result, the researchers were able to detect modularity detection 40% of the time for Eclipse JDT and 60% of the time for Hadoop, which seems little bit lower than I anticipated (about 80%). Then they explain the hardships of detecting such violations. Sometimes the violations itself might not be violation, but inherent to the design of the specific project. Also, there were some micro-refactoring that could not be matched to any violations. Lastly, If the violation only occurred once, CLIO was not able to tell whether it is violation or not, which seems to be the biggest weakness of this tool.

It makes sense to detect violations for major refactoring that occurs oftentimes. Yet, I think a tool is more useful if it can detect small-refactoring violations which do not occur quite as often as others. Personally, I did not like how they presented their data, such as table 2 or Figure 7. If you first look at these, it is not so clear to tell what information the authors try to convey. I wish that they had labeled better, or used footnotes.

Also, they did not mention whether CLIO was a tool that is developed by others, or something that they came up with in the process of their research. I wish that they included this information. Lastly, as they had recognized, it is hard to tell how CLIO generally performs, which is one of their objectives. Even if they used 10+ revision history per project, they only used 2 projects to evaluate this tool.

**Question:**

1. Table 2 seems it could be misplaced..? why would they put the table 2 far away from its explanation, which is section 4.3.1?