# Motivation

Static analysis plays a prominent role in releasing bug-free software. Despite that, essential tools suffer from well-documented usability issues. Johnson et al. found design flaws in current static analysis tools and the need for an interactive mechanism in assisting developers in fixing bugs. They interviewed with 20 participants, of which 16 are professional developers, and four are graduate students. The exciting findings are if the output of static analysis tool is user-friendly and intuitive, false positives and high number of warnings could be less problematic for a developer. Also, showing call hierarchies with which parts of code are affected by a bug, and be able to share settings with pre-defined coding standards among the team. Next, need of a web browser for reacting on the analysis output, for instance, adding a comment to a bug which goes out of context to the developer. Christakis et al. also did an empirical study on what developers want and need from program analysis. They surveyed by sending invitations to 2,000 developers within their organisation, i.e., Microsoft and received 375 responses. The resultant data is analysed and found that there are some obstacles which hinder the usage of a static analysis tool by a developer such as ’Wrong checks are on by default’, ’Too many false positives’, ’Too slow’, ’Complex user interface’. Being a user interface an obstacle for a developer along with other usability problems is noteworthy. The key takes away from the papers, as mentioned earlier, is the importance of Usability in the ongoing adaption of static analysis tools.  
  
In general, the setup of most of the recent research done in the area of Static Code Analysis is like assuming a single project in an organisation. Further, they assume a single person is working on a single project with a single tool tackling a single type of problems. Somehow, the assumptions are made singular to address a specific issue in their research. However, in practice, i.e., in the real world of software engineering, numerous people are working in teams for multiple projects at a time. Each project uses multiple tools in their software development. Even in the case of Static Code Analysis, multiple tools are used, which are each capable of addressing several types of issues in order to find more code flaws.  
  
Habib et al. did a study on static bug detectors about how the tools find many of the bugs. They found that tools used for their research are complementing each other in some bug findings. Thereby, expressed opinion that developers might want to combine the tools and so researchers could address how to reconcile the bug findings reported by multiple tools. This open challenge gives additional motivation for this thesis work.