In software engineering, code reuse is one of the standard practices in modern programming where a developer reuses some code fragment by copying and pasting either with or without any form of modification. On the one hand this has an advantage of increasing developers’ delivery time especially when the workload on the developer is much, the developer then spends less time implementing modules or functionalities that have already been implemented. On the other hand the major disadvantage of replicating code fragments is that the overall solution becomes over-large and cumbersome there by making code maintenance expensive and time consuming.

Code clone is therefore defined as the fragment of code that is akin to some other code that exist within the same or another file of a software application. Although developers’ replicate code fragments in order to reduce the burden of rewriting similar code but they end up introducing redundant code by merely copy and pasting the fragment of code at different location in the same file or in different files on the same application. Hence the needs to develop sophisticated clone detection techniques that would be able to detect the various types of clones and also protect intellectual property especially in organisations and academia.

In software engineering, there have been quite a number of researches on clone detection tools to detect and remove clones from software applications. The fact remains that clone detection in software application is not an easy task especially when dealing with very large complex systems and the cloned code is not identical to the source code i.e. the source code has been totally modified. There are a good number of techniques or approaches that have been applied to clone detection which include;

1. Text or String Based
2. Token Based
3. Program dependency graph
4. Metric Based
5. Abstract Syntax Tree
6. Hybrid Method

The String based clone detection approach which is one of the easiest clone detection techniques divides the source code into strings and then compares the string with each other to check if clone exist. The Token based clone detection technique breaks down source code into token sets and then does the comparison of each token with the other to find clones. Parse tree clone detection technique develops a parse tree of code to find clone. Also the Metric based software clone detection technique uses computed software metrics to detect software clone. Hybrid clone detection technique involves the combination of two or more existing clone detection to detect clone i.e. the hybrid techniques combines both syntactic and semantic.

In this report, the objective of the team is to design, develop and analyse a clone detection tool that would find software clones between files and then report the clone as a percentage. After researching the various techniques as said above that could be used for clone detection, the team unanimously agreed to implement the Abstract Syntax Tree approach using TXL as the implementation language. The built software would have the capacity and capability to take two source files (say Java files as an example) and then compare them at class, function and statement level to detect clones. These clones would then be outputted to a file that reports the lines where the clones exist.