**Introduction**

“*How to maintain the quality of a software?”*  Right now this question is giving sleepless night to all the code developers and eating out all the energy of a software firm. Now days every software company wants to have perfect software system that should last for long and metric is one of the measure to check the quality of the developed software system. Software metrics are designed to help developers to maintain and check the quality of a software system. It may helps in risk analysis, cost analysis etc. but only developing metrics can’t solve the problem of analyzing the quality. According to M. Olague et al [] there should be empirical validation of all the OO metrics developed to ensure their appropriate use in industries. As it might be the case that a metric may be correct from theoretical views but its of no use in practical scenario.

In our approach we developed six different metrics, four of them were based on class level calculation and two of them were based on system level but we validated seven different object oriented metrics. We introduced notion of volatility in our calculation, which is something that somehow no one still touched it. We wanted to know that how volatility is associated with quality of metrics. We defined volatility based on the number of modification that has been made to a java classes in different releases. If a java class is modified in almost all the releases then we put that class in high volatility bucket and if a java class modifies only once through out different releases, taken into account for studies we put it in low volatility metrics. We performed a case study on four consecutive version of Apache Ant to validate seven different OO metrics. Apache Ant is publically available open source. We used the information of different commits available on its website.

We validated widely used LCOM, LCOM-Henderson, RFC, Relative class size, Cohesion, MHF and AIF. Through our experiment we tried to figure out how the different metrics are correlated to each other, and we tried to find out which metrics can easily defined volatility.

Beside this we also used the historical data to build a predictive model to predict future changes based on current changes. We have taken five major releases of Apache ant ranging from 1.5 to 1.9 for our study. We have used association rule mining for generating rules and used these rules to predict future. Association rules are conditional statements that help in finding relationship between random objects in a database or information repository. It gives information that how strongly two objects are related to each other which is impossible to detect with naked eyes.

The core part of association rule is **apriori algorithm**. It is based on support and confidence. *Support* is an indication of how frequently the items appear in the database. *Confidence* indicates the number of times the if/then statements have been found to be true.

It is widely used now days in industries to find out hidden trend and pattern.

For example “*If a person buys milk then there is 60% of chances that he will buy eggs too*”. They play important role in determining customer behavior in grocery store, in shopping basket data analysis, product clustering, catalog design and store layout.

Programmers use association rules to build programs capable of Machine Learning. Machine learning is a type of artificial intelligence that seeks to build programs with the ability to become more efficient without being explicitly programmed.