1. Team Formation

Name	ID	Responsibilities/Roles
Md Tajmilur Rahman	6725201	Programming, research
Shams Azad	6768075	Statistics, writing
Amish Gala	4474325	Programming, writing, testing
Vinodkumar Raghu	7044364	Research, testing, databases

2. Type of the study

Type 2
We will examine four versions of Apache Ant (1.5, 1.6, 1.7 and 1.8) in order to identify the classes which were modified between each subsequent release. We will group these classes into three buckets:

- High Volatility (HV): the classes which changed all three times
- Medium Volatility (MV): the classes which changed two times
- Low Volatility (LV): the classes which changed only once

We will then calculate the average of specific quality metrics on each bucket.

Hypothesis: In Apache Ant, the classes which are more volatile (HV) exhibit lower quality metrics compared to classes which are less volatile (LV). We will chose a subset of class based metrics proposed by Chidamber and Kemerer (CK), Bansiya and Davis (MOOD), and those identified in Marinscu's design flaw strategies.

Based on our data collection, we will also try to predict the classes which have changed between 1.8 and 1.9, and then compare against the actual changes made between those versions.

3. Related studies

Study 1: Empirical Validation of Three Software Metrics Suites… (Olague, Etzkom, Gholston…)

This empirical study evaluates how well CK, MOOD, and QMOOD metrics are able to predict faults for a particular software program which was built using agile techniques. The paper is relevant since it provides a foundation for each individual metric, and furthermore because the authors analyze how effective they are in predicting faults of the given software system over multiple releases.

Study 2: Detection Strategies: Metrics-Based Rules for Detecting Design Flaws (Marinescu)

This study aims to group existing metrics into sets which, when taken together, are better equipped to predict design flaws in a software system. This study used two subsequent versions of a medium size business application (93-115KLOC), and thus is a good reference for our study since we will also look at subsequent versions of a given project. We may try to analyze sets of metrics together to see if we are able to detect design problems in Apache Ant.

Study 3: Diagnosing Design Problems in Object Oriented Systems (Trifu, Marinescu)

This paper tries to makes a clear distinction between structural problems and structural symptoms, and presents a novel, causal approach to restructuring object-oriented systems.

4. Projects

4.A the versions/revisions that you selected

- Apache Ant versions 1.5, 1.6, 1.7, 1.8 and 1.9
- Link: https://fisheye6.atlassian.com/browse/ant
- Code can be retrieved via branches for the first 4 versions, and then ANT_190 in Tags for the latest (or potentially HEAD).

4.B the actions you took in order to successfully compile the projects in Eclipse

1. Clone the ant-core code
2. Create a new project Eclipse project from the ant buildfile (build.xml)
3. Import a bunch of external libraries (see appendix):

4.C the presence of complementary software artifacts that you will use in your study (e.g., bug reports)

- SVN commit log (obtained via `% svn log <ant repo> -v > ant_core_history.txt`)
- We may access the Apache Ant bug database in order to obtain further details for particular bugs (https://issues.apache.org/bugzilla/)

4.D additional tools that you may use for extracting information (e.g., applied refactorings)

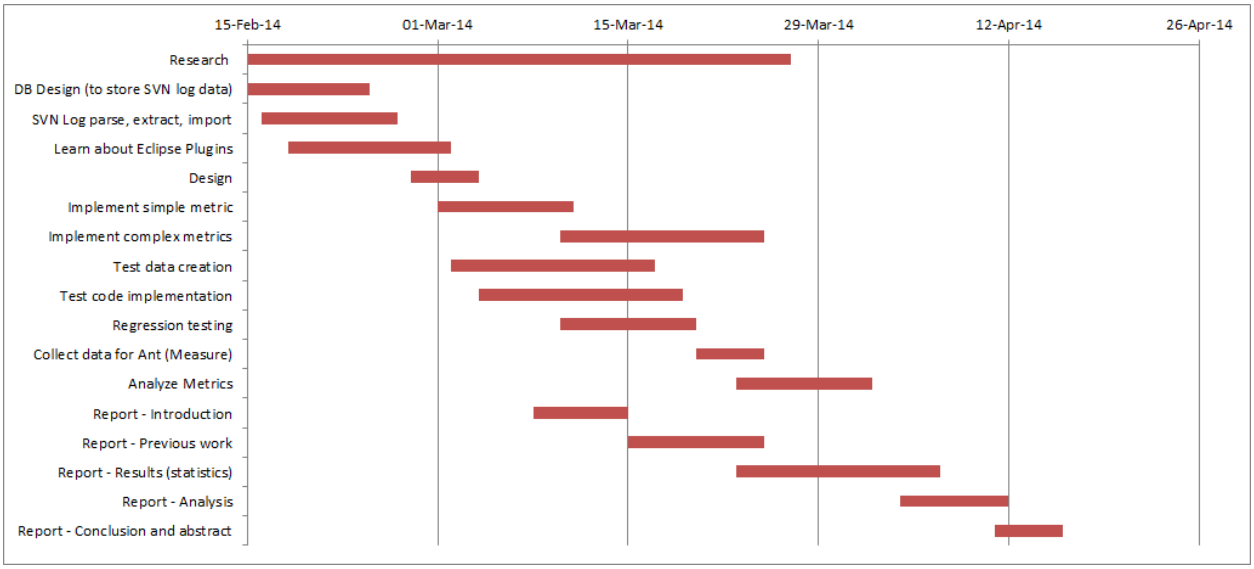
- Eclipse plugins to help identify static code issues (FindBugs)
- Visualization plugins (ObjectAid)
- RStudio for statistical analysis

5. Metrics

The following list of metrics is interesting to us in order to classify and identify the quality attributes for a given class. This list may grow / shrink depending on available time.

Number of Methods (QMOOD) [4]	Weighted Method Count (WMC from CK) [5]
Class Interface Size (QMOOD) [4]	Depth of Inheritance Tree (DIT from CK) [5]
Tight Class Cohesion (Cohesion) [3]	Response For a Class (RF from CK) [5]
Access to Foreign Data (Marinescu) [3]	

6. Resource planning



See excel file “Gant Chart.xlsx” (under **milestones** in our repo) for details on who will champion each subtask.

References

[1] H.M. Olague, L.H. Etzkorn, S. Gholston and S. Quattlebaum, "Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Process," IEEE Trans. Software Eng., vol. 33, no. 6, pp. 402-419, June 2007

[2] R. Marinescu, "Detection Strategies: Metrics-Based Rules for Detecting Design Flaws," Proc. 20th IEEE Int',l Conf. Software Maintenance, 2004

[3] A. Trifu and R. Marinescu. Diagnosing design problems in object oriented systems. In Proceedings of the Working Conference on Reverse Engineering (WCRE), Pittsburgh USA, 2005

[4] J. Bansiya and C. Davis, “A Hierarchical Model for Object-Oriented Design Quality Assessment,” IEEE Trans. Software Eng., vol. 28, no. 1, pp. 4-17, Jan. 2002.

[5] S. Chidamber and C. Kemerer, “A Metrics Suite for Object-Oriented Design,” IEEE Trans. Software Eng., vol. 20, no. 6, pp. 476-493, June 1994.

Appendix / Archive

Links to all external JARs necessary to have Apache Ant (core) imported into Eclipse without build errors

1. https://java.net/projects/javamail/pages/Home#Download_JavaMail_1.5.1_Release
2. http://commons.apache.org/proper/commons-bcel/download_bcel.cgi
3. http://commons.apache.org/proper/commons-logging/download_logging.cgi
4. http://archive.apache.org/dist/logging/log4j/1.2.17/
5. http://mavenhub.com/mvn/thirdparty-releases/com.sun.media/jai-codec/1.1.3#availableVersions
6. http://download.java.net/media/jai/builds/release/1_1_3/ (the zip file at the bottom)
7. http://clarkware.com/software/JDepend.html#download
8. http://commons.apache.org/proper/commons-net/download_net.cgi
9. http://sourceforge.net/projects/jsch/files/jsch.jar/0.1.50/jsch-0.1.50.jar/download
10. http://www.netrexx.org/downloads.nsp
    a. Put ecj-4.2, NetRexxC, and NetRexxF .jars
11. http://mirror.its.dal.ca/apache/xalan/xalan-j/binaries/
12. http://xerces.apache.org/mirrors.cgi#tools (bottom link XML Commons Resolver 1.2)
13. https://commons.apache.org/proper/commons-bsf/download_bsf.cgi
14. http://archive.apache.org/dist/jakarta/oro/binaries/