



THE AUSTRALIAN NATIONAL UNIVERSITY

FACULTY OF ENGINEERING AND INFORMATION TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

## *Project Plan*

# Variation of Defects and Complexity in Multiple versions of Open-Source Projects

**Supervisors:** Dr. Clive Boughton

**Student:** Kshitij Shukla (u4513469)

## 1. Background

As most of the companies spend too much of money in testing because they were unable to predict defects in the software. Card and Glass [1] in 1990 realised this and they came up with a design complexity Module. They argued that the design complexity measures are indicator/predictor of decision count (cyclomatic complexity), system complexity (executable lines of code), and errors (discovered from system tests)[3,4]. In their study they came up with the correlation between Design Complexity and the pre delivery defects. According to their research result:

$$\text{Number of Defects} = 0.4 \text{ Design Complexity} - 5.2$$

The variables that are of concern in the project are:

1. Procedural Complexity (Decision Count)
2. Structural Complexity (Fanout)
3. Data Complexity (Number of classes parameter)
4. System Complexity (Data Complexity + Structural Complexity)
5. Number of Defects found

All the work by Card and Glass is for Structural language Software. Chidamber and Kemerer [2] in 1994 continued the work by Card and Glass for object oriented softwares. They defined Coupling between Objects (CBO) which is the FanOut equivalent of Structural oriented Software. According to them, CBO for a class is a count of the number of other classes to which it is coupled.

## 2. Project Description

The project is to confirm how to predict the number of post delivery Defects and also to predict the maintenance effort required. The projects that will be considered for the calculation of the required variables are taken from sourceforge.net. In order to help ensure some consistency among the projects that we chose to study and to reduce the influence of at least some variables we define several characteristics to act as guidelines are follows:

1. Active Projects (Activity percentile > 90%)
2. Developed using java programming language
3. Development status is: Production/Stable
4. High number of downloads > 50,000
5. Significant error reports (error reports shown in bug tacking system)

For the purpose of getting quantitative data for our analysis, the various versions of (downloaded) java source code and binary files were examined using metrics tools such as JHawk, Jstyle and Chidamber and Kemerer Java Metrics (CKJM) [5,6,7]. The results from these tools were reported as measurements for all the system at the application, class and methods level. These results are kept in repository for further reference in the data analysis phase. In the data analysis phase of the project statical analysis tools like Rattel will be used to study the pattern/trend/ correlation between the various metrics and actual reported defects in the system.

Main Task:

- Confirming how to predict the number of post delivery defects and be able to predict the maintenance effort.
- Complexity with version: Increases or Decreases?
- Defects with version: Increases or Decreases?
- If complexity increases or decreases what is the effect on Number of defects.

### 3. Project Plan/ Schedule

As nature of Project is basically Research oriented so one of the major part of project time is given for the verification of the tool to be used as it will be having a major effect on the results.

The following table contains the estimated schedule of the project as it goes through the semester. This schedule is there to help and observe the status of project at particular time instance. As research projects have considerable amount of uncertainty in them to achieve goal so the following estimates can change little bit with the course of time.

Week	Date	Milestone
1-2	23 <sup>rd</sup> February – 8 <sup>th</sup> March	Get an idea about the project and work done in the related filed.
3	9 <sup>th</sup> March – 15 <sup>th</sup> March	Install various Metrics Calculating Software
4-8	16 <sup>th</sup> March – 12 <sup>th</sup> April	Verify the Metrics Software for Correctness.
9	13 <sup>th</sup> April – 19 <sup>th</sup> April	Download different versions of various Active Open Source Software from Sourceforge.net
10-11	20 <sup>th</sup> April – 3 <sup>rd</sup> May	Collect the appropriate measures for each version in accordance with the Object Oriented equivalent of the Card & Glass work.
12-13	4 <sup>th</sup> May – 17 <sup>th</sup> May	Analyse the results for correlation between design complexity and reported defects to determine how this relationship changes with version.
14-16	18 <sup>th</sup> May - 7 <sup>th</sup> June	Finalize Documentation and prepare for presentation

## Reference:

- [1] David N.Card and Robert L. Glass. Measuring Software Design Quality. Prentice Hall.
- [2] Shyam R. Chidamber and Chris F. Kemerer. A metrics suits for object oriented design. IEEE transactions on Software Engineering, Vol. 20, No. 6. June 1994.
- [3] Normi Sham Awang Abu Bakar and Clive Boughton. Using a combination of measurement tools to Extract metrics from open source project. Software Engineering and Applications Proceeding. November 16 – 18, 2008. Orlando, Florida, USA.
- [4] Normi Sham Awang Abu Bakar and Clive Boughton. The effect of software design complexity on defects in open source system. 2<sup>nd</sup> Int. Doctoral symposium on Empirical software Engineering (IDOESE 2007) page 104.
- [5] Denis Kozlov, Jussi Koskinen, Markku Sakkinen and Jouni Markkula. Assessing maintainability changes over multiple software releases. 2007 John Wiley & sons, Ltd. 14<sup>th</sup> September 2007.
- [6] Ligu Yu, Stephen R. Schach and Kai Chen Mesuring the maintainability of open source Software. Empirical Software Engineering, 2005. 2005 International Symposium on 17-18 Nov. 2005 Page(s):7 pp
- [7] Rudiger Lincke, Jonas Lundberg and Welf Lowe. Comparing software tools. Proceedings of the 2008 international symposium on Software testing and analysis. Session: Metrics and threads Pages 131-142. 2008. ACM.