# Reallocation of Resources during Releases Improves Cyclomatic Complexity

Md. Tajmilur Rahman£

£Department of Computer Science and Engineering, Concordia University, Montreal Quebec, Canada

# Abstract

I will write it later in the end.

# Introduction

Your introduction is too general, please work with the introduction I have started below and our discussion that is in your status file.

Software projects are notorious for going over budget and schedule. Before a major release, there is often a rush period where many tasks are finished. This “rush to release (RTR)” can be prompted by external forces, such as decision by management to include new features in a release or to release earlier to beat a competitor. Alternatively, the rush may simply be due to inappropriate or unrealistic scheduling. Regardless of the causes, the rush to release stresses developers and often requires developers to work on unusual, high priority areas of the system. In this paper, we study how RTR effects project organization and the introduction of technical debt. Our main research questions are as follows:

1. Do developer work on different areas of the system around the time of release?
2. Are there certain areas of the system that receive increased attention (ie do developers focus on a smaller set of files around releases).
3. Do the areas of code that are modified around the time of release have higher defect densities than code that is modified during normal development?

Please do a literature search on “software releases”, “software rush to release” etc

Please ignore complexity in general and cyclomatic complexity in particular. Customer reported defects is the only outcome of importance, complexity measures are useless, you may as well measure file size.

Software development is a process where developers make computer programs to make it work as the customer demands. There are many ways to accomplish these demands. Some are traumatic, some are good and there are very few ways that are absolutely perfect. According to the way the development team is following software development projects often suffer trauma regularly. There are many kinds of traumas we can mention here. Sometimes developers follow shortcuts to the development to meet the deadlines that introduce technical debt. If this technical debt is not repaired, quality of the system will suffer in the long run. If a project owner or leading developer who owns a vital or a big part of codebase, also then some problems may arise for the other developers to take over those parts of the project which may sometimes increase the chance of hidden bugs. Some times management reorganizes the developers on a company’s projects, with the result that developers move to codebases for which they have less experience. The reorganization introduces new perspectives and expertise that can lead to innovation; however, it can also result in a drop in productivity and the unnecessary re-writing of large portions of the system that the new developers do not understand. In this paper we are trying to look into this particular traumatic behavior and will try to understand what impact gets put on the cyclomatic complexities of the files been worked on in a particular release if there is no proper re-allocation of resources (developers) happens among the development teams in a software development industry. We also answer the following research questions in this work:

1. Which way the resources in a software development industry should be re-allocated among the software development teams?
2. What is the impact on codebase if resources among the development team are not properly re-allocated during release period.

Very few research works have been performed regarding the re-allocation of resources. Robert van Engelen [1] worked for similar kind of a research to understand the resource allocation dynamics across the software projects. He mainly tried to reallocate development resources amongst projects for increasing the satisfactory level of consumer or customer while we are focusing on the impact on codebase like the complexity of script files. Robert proposed a project entropy metric in his work to understand if there is any limit for a particular reallocation does not lead to user satisfaction. Here entropy is to represent disorder and chaos to understand degradation of software and its inherent complexity. In his work resources may not just be the developers but also can be any other resources necessary for a software project development.

This paper is organized as follows. In Section II, we describe some background and motivation. Section III will tell us about the ownership of files and ownership of a set of files or a directory. We will show how to understand how native a codebase is to a developer or a development team. In Section IV, we will try to show some analysis to determine reallocation has been performed in a release or where reallocation needs to be performed. What changes in nativeness ∆ƞ occurs after the reallocation. Section V will give us the result to show how change in ∆ƞ is correlated to the impact on the change in the complexity in codebase. Finally section VI will give us an idea of our future work and followed by the section VII Conclusion.

# Background and Motivation

…

# Ownership and Native-Code

…

# Research and Analysis

…

# Results and Discussion

…

# Future Work

…

# Conclusion

…

# References

1. Robert van Engelen, Subhajit Datta “*Project-entropy: A Metric to Understand Resource Allocation Dynamics across Software Projects*”