# Title: Relating change complexity and system complexity

Name

# Abstract

Write this last.

# Introduction

Traditional complexity measures are normalised on or require a whole

file or system and were not designed to measure the complexity of code

fragments. The goal of this paper would be to evaluate which measures

are the best indicators of the complexity of an evolving system. One

point to note is that the complexity of a change can be very

different from the complexity of the files or system that contain a

change (e.g., a change to many simple files might be as complex as a

change to one difficult file). Change complexity is related to, for

example, "how hard is it to understand and review a change" vs system

complexity, which measures for example, "how hard is to understand and

potentially modify the system". ..

Furthermore, traditional complexity measures are not really

measuring the complexity of the system but simply the size of the files that make up the system. As a result, they do not add any additional information beyond how large a file is.

In contrast, all of the change complexity measures are based upon the notion that changes that are farther

from each other or involve multiple entities are more complex than those closer

together involving fewer entities. In total, we use seven measures of change

complexity: the churn (or size), the number of modified files in a diff, the

number of diffs per review, the number of and distance between contiguous

change blocks within a diff (\ie hunks), the directory distance between these

files, and depth of indentation of a change (Hindle2008ICPC).

In this work, we answer the following research questions:

1- Is there a relationship between traditional complexity measures and change complexity measures?

2- Does the change complexity measure at version (i) of a system relate to the difference between the traditional complexity measure at version (i+1) and version (i).

CCMv(i) ~ TCMv(i+1) – TCMv(i)

3- Does the traditional complexity measure at version (n) relate to the sum of all change complexity measures of the system?

TCMv6 ~ ∑0-6 CCMv(i)

The paper is organized as follows. In Section II, we describe … In Section III, we …

# Measures

# Background/Literature

CMetrics measures the size and complexity of C files. In the following section we present a brief description of each metric.

1. Metrics related to lines of code.

- Source Lines of Code (SLOC):

To define SLOC we use the definition given by Conte [Conte 1986]:

A line of code is any line of program text that is not a comment or blank line, regardless of the

number of statements or fragments of statements on the line. This specifically includes all lines containing program headers, declarations, and executable and non-executable statements.

* Lines of Code (LOC)

Lines of code refers to the total number of lines in each source code file including comments and blank lines. It is a straightforward measure and can be calculated using many available tools.

* BLANK

Blank is a count of the number of blank lines.

* COM.L

Number of lines that are exclusively comments (no code).

* COM.N

Number of comments in the file (a comment can be multiline).

1. Metrics related to complexity

* McCabe’s Cyclomatic Complexity (TOTCY)

McCabe’s Cyclomatic complexity is one of the earliest complexity measures developed by Thomas J. McCabe in 1976. It directly measures the number of linearly independent paths through a program’s source code. Any program can be represented as a graph with the simplest element being a flat series of statements with no conditions, loops or branches. [[1](#Her10)] For a graph G with n vertices, e edges and p exit points, the complexity v is defined as follows:

v(G)=e-n+2p

It is worth mentioning that the minimum value for the cyclomatic complexity metric is 1, which corresponds to the flat series of statements with no bifurcations or loops. Every additional region in the flow graph would increase the Cyclomatic complexity by one unit.

* MAXCY

Maximum McCabe’s cyclomatic complexity. (between all functions).

* MINCY

Minimum McCabe’s cyclomatic complexity.

* AVGCY

Average McCabe’s cyclomatic complexity.

* MEDCY

Median McCabe’s cyclomatic complexity.

-Halstead’s Complexity Measures

Halstead’s complexity metrics rely on the assumption that programs should be viewed as expressions of languages both programming and written. It relies on the premise that there are mathematically sound relationships among the number of variables, the complexity of the code and the type of programming language statements used.

* H.LEN

Halstead’s length.

* H.VOL

Halstead’s volume.

* H.LEVEL

Halstead’s level.

* H MEN.D

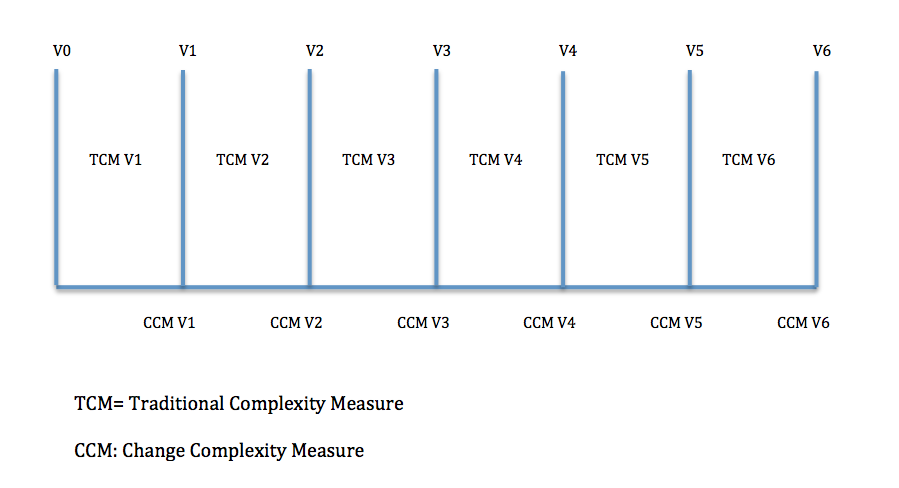
Halstead’s number of mental discriminations.

# Change Complexity Measures

Copy from my thesis and elaborate

# Methodology

In figure 1 we illustrate a hypothetical system, which undergoes numerous changes. The vertical lines represent each version. During each version's development, the change complexity measures are assessed. In between versions, traditional complexity measures are computed. These are denoted by CCM and TCM respectively.



[Add mathematical equations here].

# Results

# Discussion

# Conclusion