# Title:

Software Complexities: Do System and Change Complexity Relate?

# Abstract

Write this last.

# Introduction

Write the introduction first. It scopes the document and tells the reader what you will be talking about. It is basically a mini version of the paper from motivation all the way to results. You will also need to tell us how the paper is structured.

In this work, we answer the following research questions:

1. One short sentence that summarizes the question
2. You should have 1 to 3 questions, no more

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

Traditional complexity measures are normalized on or require a whole file or system and were not designed to measure the complexity of code fragments. ~~The goal of this paper is to evaluate which measures are best indicators of the complexity of an evolving system as well as relate traditional complexity measures with change complexity measures~~. Therefor, 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, ``how hard is it to understand and review a change" vs system complexity, which measures for example, "how hard is it to understand and potentially modify the system".

Traditional complexity measures (e.g. McCabe) are normed on or require a whole file or system and were not designed to measure the complexity of changes and code fragments. Furthermore, traditional complexity measures do not really measure 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. The goal of this paper is to evaluate the correlation between system complexity measures and change complexity measures of an evolving system.

We have the following research questions:

1- Does the difference of traditional complexity measures before and after a change correlate with change complexity measures?

2- Does the weighted sum of change complexity measures up to a version correlated with the traditional complexity measures at that version?

3- Do the traditional measures calculated on only the files that had changed at a certain commit correlate with the change complexity measures at the last 1000 changes?

# Background and Motivation

Tell us why the problem is important. If you solve the research problem, who will care? If nobody will care, then don’t do the research. Maximum impact.

You will also summarize each of the papers you reference in this work in a paragraph. The paragraph must clearly show how the paper DIRECTLY relates to the work being done. If it doesn’t related directly, cut it!

- Israel :

(Traditional)

Traditional complexity metrics require the system as a whole and were not designed to measure the complexity of code fragments. McCabe’s metrics directly measures the number of linearly independent paths through a program's [source code](http://en.wikipedia.org/wiki/Source_code). Syntactic complexity metrics that are exclusively based on the structure of the program and the properties of the text (for example, redundancy of operators and operands, as Halstead’s metrics do), do not add any more information than lines of code do. [Isreal]. Furthermore, multiple studies have shown that these measures correlate very strongly with lines of code and may not provide additional information about the complexity of the system. [54,85,61].

(Change C)

As previously stated, measuring the complexity of a change differs from measuring an entire system. For example, a change to numerous simple functions across multiple files may be as difficult as changing the code contained within a single complex function. [Peter].

In this section, we provide a background on the various measures of system and change complexity.

## System (Traditional) Complexity Measures

### Metrics Related to 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 of 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.
* 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: 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).

### Metrics Related to Complexity

* **McCabe’s Cyclomatic Complexity:** 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. 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.

* Totcy: Total McCabe’s cyclomatic complexity (sum of all functions).
* Maxcy: Maximum McCabe’s cyclomatic complexity between all functions.
* Mincy: Minimum McCabe’s cyclomatic complexity between all functions.
* 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

Since the goal of each measure is the same, to assess the complexity of a change, we expect some of our measures to be highly correlated. In the interest of parsimony, we will select the simplest set of measures that adequately captures change complexity.

* Files and diffs (Mods): We count the number of files contained in a commit. The more files that change, the larger the affect proportion of the system. Commits with a large number of files will likely be difficult to understand.
* Change blocks (Hunks): We measure the number of contiguous change blocks, or hunks, and the distance between these blocks in a modified file and sum them across the entire review. Contiguous changes are likely easier to review than changes that are further apart and that are potentially in multiple functions within a file.
* ~~Directory distance: We measure the directory distance between the files in a commit. The premise for this measure is that files that preform similar functions are typically closer in the directory hierarchy than files that perform dissimilar functions [Bowman 1999ICSE], thus the directory structure loosely mirrors the system architecture [RigbySubTOSEM]. The distance of two files in the same directory is zero, while the distance for files in different directories is the number of directories between the two files in the directory hierarchy. We expect that changes that involve files that are far apart will crosscut the functionality of the system and be more complex to review.~~
* Indentation: [Hindle2008ICPC] creates a complexity measure that can be used to measure the complexity of the entire system or of a change by examining the level of indentation on source lines. We calculate their measure for reviews. They found that the strongest predictors of complexity were the sum and standard deviation of the indentation of source lines.

# Methodology and Data

Describe your methodology. Creswell is a good place to look for ways to discuss your methodology.

Describe where your data came from, how it was collected, any limitations the data has, etc

A quantitative research methodology was used for this empirical study where the objective was to develop models pertaining to our research questions. The choice of this method is key as it provides a connection between the empirical observation and mathematical expressions of the quantitative relationships.

The data used for this study was provided by Dr. Peter Rigby, namely from the Apache HTTP Server Project written in C. All measures were calculated at a particular version (commit) and the data was retrieved from June 18, 1999 – March 3, 2011. SQL scripts were written to extract data for each of the research questions using PostgreSQL 9.3. The resulting data was then imported into RStudio for further analysis. Spearman’s rank correlation coefficient matrix was then deduced for each measure.

# Results and Discussion

Start each result section with the header of the question to remind people why you have the result. Describe the result. Then discuss the impact the result has.

# Threats to Validity

Here you talk about any problems you’ve had while doing the study.

# Conclusion

Tell ‘em what you told ‘em