Do Programming Languages Influence the Impact of Software Changes?

Shaista Ghafoor
Dept. of Computer Science
University of Sargodha
Sargodha, Pakistan

Javed Ferzund
Dept. of Computer Science
University of Sargodha
Sargodha, Pakistan

Bushra Jamil
Dept. of Computer Science
University of Sargodha
Sargodha, Pakistan

ABSTRACT
During a software development process, changes happen in almost every phase: requirements, design implementation, and maintenance. Software-change impact analysis, or simply impact analysis (IA), has been recognized as a key maintenance activity. IA aims at estimating the potentially impacted entities of a system due to a proposed change. In this paper, we present a study to investigate the role of programming languages in change impact analysis. We try to find whether changes made in different language programs have same impact on different entities or not same. In this study IA is based on number of files impacted, number of revisions impacted, number of developers involved and changes made per hour.

Keywords
Software Maintenance, Software Changes, Bugs, Programming Language.

1. INTRODUCTION
Change impact analysis is an important activity in software maintenance. The purpose of impact analysis is to study the software entities that are affected or impacted by a given change. A single change may affect a single file or a group of files. Within a single file, a change may affect a single code location or multiple code locations. So, a single change can create challenges for the testing team. A tester will have to test all the possible impacted source code locations which can be in different files and different modules.

The knowledge of the impacted files and modules is not known unless project history is studied. We can extract this information by mining software repositories. Software repositories hold all the information related to a particular change like when a change was made, who made that change, which files were modified, how many lines were affected, which lines were affected and the reason for a change. This information is easily accessible by using configuration management systems like CVS and SVN.

A lot of work has been done on change impact analysis and still more work is going on. Researchers have tried to identify different change patterns like co-changed modules, co-changed files, co-changed lines and time periods when changes are more risky. Some other researchers have studied the spectrum of change impact and developed different prediction models for changed entities and the developers involved. To our knowledge, no work exists on the comparison of different programming languages. In other words, how changes impact in programs written in different programming languages.

In this paper, we present an empirical study on change impact analysis. Three languages are selected for impact analysis including C, C++ and JAVA. To eliminate the differences in project domains, we have selected a single project, parts of which are developed in C, C++ and JAVA. Mozilla is used as a case study, which is a large project having long development history. Mozilla is an internet suit that comprises a browser (Firefox), an email client (thunderbird) and other tools regarding editing and authoring of web pages. We obtain the required data from the Mozilla project repository. We study the impact of changes in different language programs based on the number of files affected, number of revisions affected, number of developers required to implement a change and the number of changes implemented in one unit time. We have established the following four hypotheses to fulfill our research objectives:

Research Hypotheses:

$H_0$1: Average number of files affected by a single change is similar in different languages.

$H_0$2: Average number of revisions affected by a single change is similar in different languages.

$H_0$3: Average number of developers to implement a single change is similar in different languages.

$H_0$4: Average number of changes implemented in one hour is similar in different languages

Remaining paper is distributed into the following sections: In section II we discuss the related work and in section III the methodology is discussed. Results are described in section IV and finally we conclude the paper in section V.

2. RELATED WORK
Change Impact Analysis is a collection of techniques for determining the effect of a change. Different techniques are used for impact analysis like conjunctive approach and disjunctive approach. The result of cvs repositories is used to understand developer role on software projects like interaction frequency. It is defined as interaction between two developers based on frequency of email correspondence, frequency of co-editing, frequency of task sharing, and so on [11]. Mining software repositories can provide information about what classes changed together. Zhang and Zhao studied the possible impacts of a proposed change in AspectJ programs [12]. Software engineers use their knowledge about the dependencies in the software architecture to properly perform impact management. Souza and Redmiles studied how developers manage dependencies and changes [9].
Canfora and Cerulo used CVS and Bugzilla to study impact of change requests [3]. The information retrieval algorithm used source files and developers for predicting impacted source files by new change request and gave list of candidate developers for resolving new CR. Mockus and Weiss [7] compute experience of developer by checking out the number of changes a specific developer has made to change software. While Mockus and Herbsleb introduced a visualization tool Expertise Browser (ExB) that is used to find the desire expert for particular artifact of code, or to find profile of expertise of specific person, group of people or organization [6].

Minto and Murphy [5] introduced Emergent Expertise Locator (EEL) tool that is used to recommend ranked list of expert team within their development environment. They used this tool on Eclipse's, Firefox's, and Bugzilla's historical data. Used two matrices, one was file dependency, and second was file authorship matrix that produced third matrix of expertise. Wong et al. developed a tool Clio for predicting coordination structures of change requests. This framework predicted related files and owner of these files that can be changed to fulfill modification request, based not only on historical data but also on the design structure [10].

Cerulo and Canfora linked new CR descriptions with revision files impacted by the past similar fixed CRs. They used textual similarity for this purpose. They also showed that mining software repositories are not just used in software evolution but can also used for impact analysis [4]. Siy et al. recorded modified files with respect to the developer who changed them. This work is done by splintering CVS change data of each developer. They showed that most of the developers worked on same files/directories [8]. Through this way one can easily know about individual developer expert area.

3. METHODOLOGY

Any study on change impact analysis requires data from software repositories. Some data is transparent and other is hidden in these repositories. The hidden data is mined to extract useful information for further analysis. This study is completed in five phases: CVS Checkout, Data Extraction, Data Storage, Data Transformation and Statistical Analysis.

CVS Checkout:

A CVS Client [1] can be used to check out source code of a project managed by concurrent versioning system. CVS client provides a checkout command that uses the path to the repository for access. Using CVS checkout, we downloaded the source code of browser, plugins, mail and modules from the Mozilla repository [2]. Table 1 shows the number of components for each programming language.

Table 1. Components for different Languages

Language	Number of components
JAVA	4
C++	4
C	3

Data Extraction:

Whenever a change is made to a file, CVS creates a new revision for that file. CVS also records some information about each change like date and time of change, author of change, lines of code added or deleted and a brief description about the change including the change request id/bug id. This information can be extracted using the log command of CVS. We executed the log command on the root directory and stored the log information into a text file. This text file is further processed to store information into a database.

CVS also provides the facility to take difference between two revisions. The difference shows the lines of code added, deleted or modified between two revisions. We executed the CVS diff command and obtained the difference of all consecutive revision pairs. This information was stored into a text file for each component separately.

Data Storage:

The text file containing the log information was processed and information relevant to individual files was stored into a MySQL table named "Files". The attributes of the Files table are shown in Table 2. Information relevant to revisions of a file was stored into a MySQL table named "Revisions". The attributes of the Revisions table are shown in Table 3.

Table 2. Files table description

Field Name	Description
rcscode	A unique id for each file
filename	File name consisting of complete path to root directory
head	Most recent revision
totalrev	Total no of revisions for this file

Table 3. Revisions table description

Field Name	Description
revcode	A unique id for each revision
rcscode	Foreign Key for Files table
revision	Revision no
rdate	Revision Date
rtime	Revision Time
developer	Developer who made the change
comment	Information about the change
buglist	Bug ids extracted from the comment

The text file containing the difference was processed and the information on changed code was stored into a MySQL table named "Difference". The attributes of the difference are shown in Table 4.

We applied different SQL queries to extract data from Tables 2, 3 and 4. These queries were designed to obtain the data relevant to our research hypotheses. These queries were executed separately for each language programs.

- The first query is designed to find out total number of files affected as a result of a particular change. The information was grouped on the basis of change id.
- Second query is designed to determine the number of revisions that have been made to fix a particular bug/change request.
- Third query is designed to find out the number of developers who worked on a single change.
- Fourth query was designed to determine the number of changes implemented in one hour.

Table 4. Difference table description

Field Name	Description
rcscode	Foreign Key for Files table
revision	Revision no
changedelta	Information on portions of source code that are changed
changecode	The code changed as a result of change request

Transformation:
The queries mentioned in the previous sub-section were executed and the results were transformed into CSV format. The transformed data was used for statistical analysis.

Statistical Analysis:
Mann Whitney test is selected to test the established hypothesis. It is a non-parametric test used to compare the medians of two samples of independent observations. Histograms are also used to compare the frequency distribution of different samples of observations.

4. RESULTS

In order to test the Null Hypotheses established in section 1, we have split each hypothesis into three sub-hypothesis, one sub-hypothesis for each pair of languages. The three sub-hypotheses for $\mathbf{H_0 1}$ are given below:

$H_0$1A: Average number of files affected by a single change is similar in C and CPP programs.

$H_0$1B: Average number of files affected by a single change is similar in C and JAVA programs.

$H_0$1C: Average number of files affected by a single change is similar in CPP and JAVA programs.

Similarly, sub-hypotheses were established for $\mathbf{H_0 2}$, $\mathbf{H_0 3}$ and $\mathbf{H_0 4}$. Mann Whitney test was used to accept or reject the Null Hypotheses. The variable 'Number of Files' is selected as "Test Variable" and 'Programming Language' is selected as "Group Variable", for $\mathbf{H_0 1}$. Similarly, the variable 'Number of Revisions' is selected as "Test Variable" and 'Programming Language' is selected as "Group Variable" for $\mathbf{H_0 2}$. The variable 'Number of Developers' is selected as "Test Variable" and 'Programming Language' is selected as "Group Variable", for $\mathbf{H_0 3}$. Similarly, the variable 'Number of Changes' is selected as "Test Variable" and 'Programming Language' is selected as "Group Variable", for $\mathbf{H_0 4}$.

After applying the Mann Whitney test, following results have been found for $\mathbf{H_0 1A}$.

Ranks

	languagegrp1	N	Mean Rank	Sum of Ranks
files	C	123	130.49	16050.00
	CPP	123	116.51	14331.00
	Total	246		

Test Statistics[a]

	files
Mann-Whitney U	6705.000
Wilcoxon W	14331.000
Z	-1.642
Asymp. Sig. (2-tailed)	.101

a. Grouping Variable: languagegrp1

Since (p-value= 0.101> 0.05 =α), the null hypothesis is accepted.
Conclusion: At α = 0.05, there is enough evidence to conclude that there is no statistically significant difference in the number of files affected by a change in C and CPP programs.
For $\mathbf{H_0 1B}$, following results have been obtained.

Ranks

	languagegrp2	N	Mean Rank	Sum of Ranks
files	C	123	136.98	16849.00
	java	123	110.02	13532.00
	Total	246		

Test Statistics[a]

	files
Mann-Whitney U	5906.000
Wilcoxon W	13532.000
Z	-3.473
Asymp. Sig. (2-tailed)	.001

a. Grouping Variable: languagegrp2

Since (p-value= 0.001< 0.05 =α), the null hypothesis is rejected.
Conclusion: At α = 0.05, there is enough evidence to conclude that there is statistically significant difference in the number of files affected by a change in C and JAVA programs.
For $\mathbf{H_0 1C}$, following results have been obtained.

Ranks

	languagegrp3	N	Mean Rank	Sum of Ranks
files	CPP	123	122.77	15101.00
	java	123	124.23	15280.00
	Total	246		

Test Statistics[a]

	files
Mann-Whitney U	7475.000
Wilcoxon W	15101.000
Z	-.209
Asymp. Sig. (2-tailed)	.834

a. Grouping Variable: languagegrp3

Since (p-value= 0.834> 0.05 =α), the null hypothesis is accepted.

Conclusion: At α = 0.05, there is enough evidence to conclude that there is no statistically significant difference in the number of files affected by a change in CPP and JAVA programs.

After applying the Mann Whitney test, following results have been found for $\mathbf{H_0 2A}$.

Ranks

	languagegrp1	N	Mean Rank	Sum of Ranks
revision	C	123	136.32	16767.50
	CPP	123	110.68	13613.50
	Total	246		

Test Statistics[a]

	revision
Mann-Whitney U	5987.500
Wilcoxon W	13613.500
Z	-3.023
Asymp. Sig. (2-tailed)	.002

a. Grouping Variable: languagegrp1

Since (p-value= 0.002< 0.05 =α), the null hypothesis is rejected.

Conclusion: At α = 0.05, there is enough evidence to conclude that there is statistically significant difference in the number of revisions affected by a change in C and CPP programs.

For $\mathbf{H_0 2B}$, following results have been obtained.

Ranks

	languagegrp2	N	Mean Rank	Sum of Ranks
revision	C	123	136.98	16848.50
	java	123	110.02	13532.50
	Total	246		

Test Statistics[a]

	revision
Mann-Whitney U	5906.500
Wilcoxon W	13532.500
Z	-3.193
Asymp. Sig. (2-tailed)	.001

a. Grouping Variable: languagegrp2

Since (p-value= 0.001< 0.05 =α), the null hypothesis is rejected.

Conclusion: At α = 0.05, there is enough evidence to conclude that there is statistically significant difference in the number of revisions affected by a change in C and JAVA programs.

For **$H_0$2C**, following results have been obtained.

Ranks

	languagegrp3	N	Mean Rank	Sum of Ranks
revision	CPP	123	124.50	15313.50
	java	123	122.50	15067.50
	Total	246		

Test Statistics[a]

	revision
Mann-Whitney U	7441.500
Wilcoxon W	15067.500
Z	-.245
Asymp. Sig. (2-tailed)	.806

a. Grouping Variable: languagegrp3

Since (p-value= 0.806> 0.05 =α), the null hypothesis is accepted.

Conclusion: At α = 0.05, there is enough evidence to conclude that there is no statistically significant difference in the number of revisions affected by a change in CPP and JAVA programs.

Since (p-value= 0.146> 0.05 =α), the null hypothesis is accepted.

Conclusion: At α = 0.05, there is enough evidence to conclude that there is no statistically significant difference in the number of developers involved in implementing a change in C and JAVA programs.

For **$H_0$3C**, following results have been obtained.

Ranks

	languagegrp3	N	Mean Rank	Sum of Ranks
developers	CPP	123	126.50	15560.00
	java	123	120.50	14821.00
	Total	246		

After applying the Mann Whitney test, following results have been found for **$H_0$3A**.

Ranks

	languagegrp1	N	Mean Rank	Sum of Ranks
developers	C	123	123.45	15184.00
	CPP	123	123.55	15197.00
	Total	246		

Test Statistics[a]

	developers
Mann-Whitney U	7558.000
Wilcoxon W	15184.000
Z	-.023
Asymp. Sig. (2-tailed)	.982

a. Grouping Variable: languagegrp1

Since (p-value= 0.982> 0.05 =α), the null hypothesis is accepted.

Conclusion: At α = 0.05, there is enough evidence to conclude that there is no statistically significant difference in the number of developers involved in implementing a change in C and CPP programs.

For **$H_0$3B**, following results have been obtained.

Ranks

	languagegrp2	N	Mean Rank	Sum of Ranks
developers	C	123	126.48	15556.50
	java	123	120.52	14824.50
	Total	246		

Test Statistics[a]

	developers
Mann-Whitney U	7198.500
Wilcoxon W	14824.500
Z	-1.454
Asymp. Sig. (2-tailed)	.146

a. Grouping Variable: languagegrp2

Test Statistics[a]

	developers
Mann-Whitney U	7195.000
Wilcoxon W	14821.000
Z	-1.467
Asymp. Sig. (2-tailed)	.142

a. Grouping Variable: languagegrp3

Since (p-value= 0.142> 0.05 =α), the null hypothesis is accepted.

Conclusion: At α = 0.05, there is enough evidence to conclude that there is no statistically significant difference in the number of developers involved in implementing a change in CPP and JAVA programs.

Based on the results of Mann Whitney test, **$H_0$4A**, **$H_0$4B** and **$H_0$4C** were rejected. It indicates that number of changes implemented in a single hour is statistically different in the studied different language programs.

Figure 1 represents the frequency distribution of number of revisions affected by a single change. First part shows the number of revisions for C programs. It is evident that most of

the changes were implemented by less than ten revisions. Very few changes required more than ten revisions and in some cases this number exceeded even 80 revisions. Similar is the case for CPP and JAVA programs as represented in second and third part of Figure1.

Frequency distribution of number of files affected by a single change is represented in Figure 2. First part shows the number of files affected in C programs. It can be seen that most of the changes affected less than 10 files. A significant number of changes also affected between 10-20 files. Very few changes affected more than 20 files. In CPP and JAVA programs majority of the changes affected less than 5 files, as depicted in second and third part of Figure 2. Very few changes affected more than 10 files.

Frequency distribution of number of developers involved in a change and number of changes made per hour are represented in Figure 3 and Figure 4 respectively.

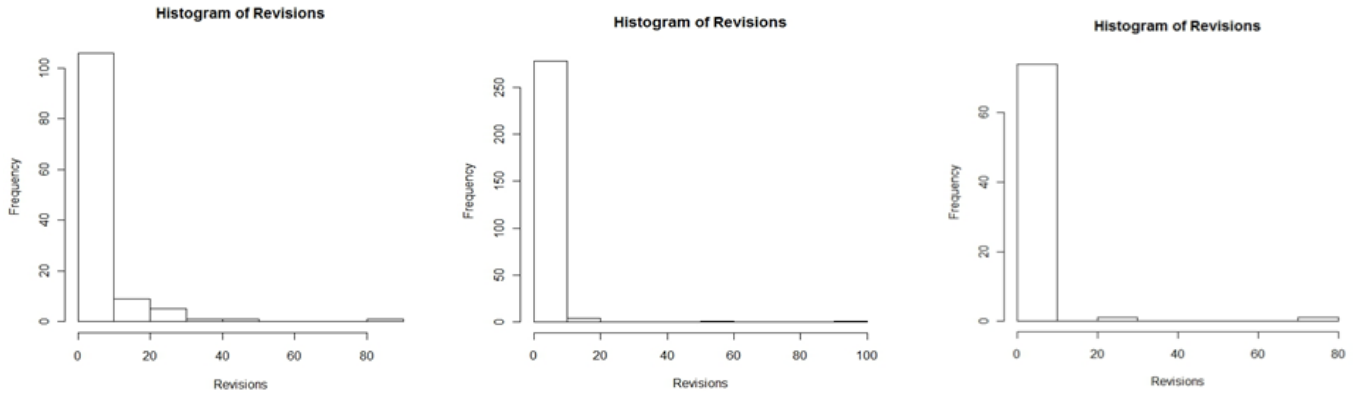

Fig 1: Histograms of Number of Revisions for C, CPP and Java Programs

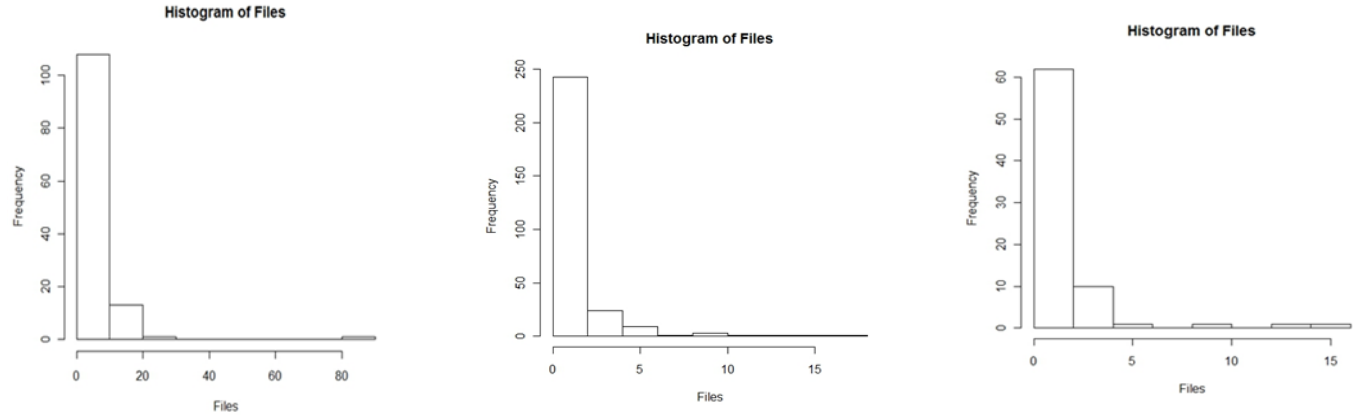

Fig 2: Histograms of Number of Files for C, CPP and Java Programs

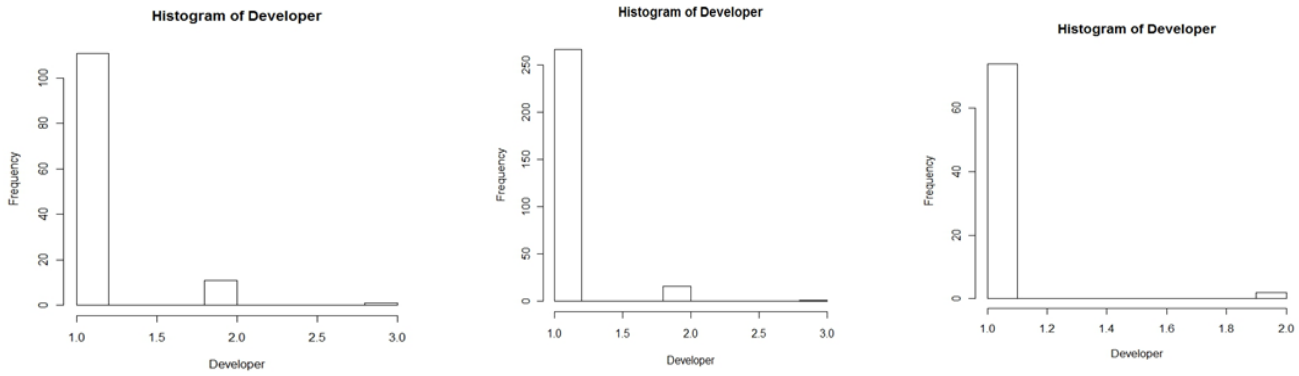

Fig 3: Histograms of Number of Developers for C, CPP and Java Programs

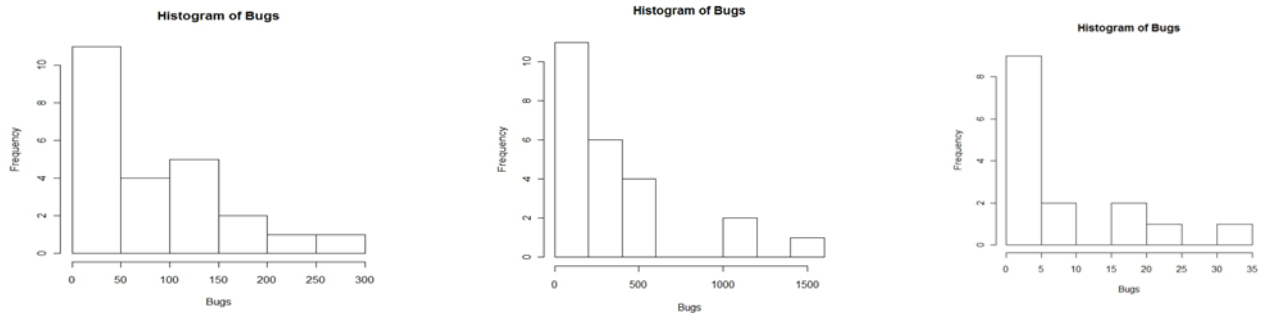

Fig 4: Histograms of Number of Bugs Fixed/ Hour for C, CPP and Java Programs

5. CONCLUSIONS

In this paper we have presented an empirical study on change impact analysis. Our objective was to investigate the influence of programming languages on the impact of changes. We established four hypotheses to satisfy our research questions. The research questions were related to number of files impacted by a change, number of revisions impacted by a change, number of developers involved in a change and number of changes made per hour.

It is found that there is no statistically significant difference between C and CPP programs for the number of files impacted by a single change. Similar is the case for CPP and JAVA programs. However, C and JAVA programs significantly differ in the number of files impacted by a single change.

In case of number of revisions impacted by a single change, statistically significant differences have been found between C and CPP programs and C and JAVA programs as well. However, CPP and JAVA programs are found similar in the number of revisions impacted by a change.

Average number of developers required to implement a change and the number of changes implemented per hour have been found similar in C, CPP and JAVA programs.

6. REFERENCES

[1] http://www.nongnu.org/cvs/

[2] http://www.mozilla.org

[3] G. Canfora, L. Cerulo, "How Software Repositories can help in Resolving a New Change Request", 2005.

[4] L. Cerulo, G. Canfora, "Impact Analysis by Mining Software and Change Request Repositories", 2005.

[5] S. Minto and G. C. Murphy, "Recommending Emergent Teams" , 2007.

[6] A. Mockus and J. D. Herbsleb, "Expertise Browser: A Quantitative Approach to Identifying Expertise", 2002.

[7] A. Mockus and D. Weiss, "Predicting risk of software changes", vol. 5(2), 2000.

[8] H. Siy, P. Chundi, and M. Subramaniam, "Summarizing developer work history using time series segmentation: challenge report," in MSR '08: Proceedings of the 2008 international working conference on Mining software repositories, 2008, pp. pages 137–140.

[9] C. R. B. de Souza and D. F. Redmiles," An empirical study of software developers' management of dependencies and changes", In ICSE'08 : Proceedings of the 30th international conference on Software engineering, pages241–250, New York, NY, USA, 2008. ACM.

[10] S. Wong, Y. Cai, and M. Marron, "Predicting Coordination Structure by Change Impact Analysis", 2010.

[11] L. Yu and S. Ramaswamy, "Mining CVS Repositories to Understand Open-Source Project Developer Roles", In Proceedings of the 4th International Workshop on Mining Software Repositories, IEEE Computer Society, Washington DC (2007).

[12] S. Zhang and J. Zhao, "Change impact analysis for AspectJ programs", Technical Report SJTU-CSE-TR-07-01, Center for Software Engineering, SJTU, Jan 2007. Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems.