Journal of Theoretical and Applied Information Technology
15th April 2023. Vol.101. No 7

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AUTOMATED CHANGE IMPACT ANALYSIS TOOL FOR SOFTWARE MAINTENANCE PHASE

P JALAJA[1], T ADILAKSHMI[2]

[1]Assistant Professor, Vasavi College of Engineering, Department of CSE, Telangana, India

[2]Professor, Vasavi College of Engineering, Department of CSE, Telangana, India

E-mail: [1]jalaja.t@staff.vce.ac.in, [2]t_adilakshmi@staff.vce.ac.in

ABSTRACT

Often, software projects are enhanced based on clients' requests or companies' policies. This process happens in the maintenance phase of the project. The maintenance phase of the Software Development Life Cycle (SDLC) occurs after the product is in production. Maintenance of software can include software upgrades, repairs, and fixes. The required changes are mentioned in the form of change request (CR) which is given to the developers who calculate the impact of changes on the software application. The effect must be assessed without modifying the software application. The change impact analysis is one such way that helps in assessing the impact.

It takes time for the developer to analyze the change request and the software application to identify the required changes. In addition, impacts of the changes need to be identified, which requires more time for the developer. And also there is a high chance of missing the impacts due to manual effort. Rather, we can use an automated tool that does the change impact analysis within less time and give more accurate results. The objective of this paper is to develop and test the automated CIA Tool (StaticPy) that performs the change impact analysis. With the help of the tool developer will have an idea of the changes to be made. This will reduce the cost and time.

So, this paper proposes a Change Impact Analysis Tool (CIAT) - StaticPy that helps to identify the impacted files, methods, fields, and elements that are affected because of the proposed changes. CIAT takes a change request and project repository GitHub link as input. Then the tool does static analysis on the given repository and forms a data structure. This data structure contains all the details of the impacted elements. From the data structure, we display the information of affected files, methods, fields, and impacted line numbers in the files. StaticPy has been developed and tested using four different software applications. The average accuracy of the tool is 97.8%.

Keywords: *Change Impact Analysis, CIA Tool, data structure, static analysis, Tokenizer.*

1. INTRODUCTION

According to the Software development life cycle, once the developed software is tested properly it goes to production. Once the software is up and running, it often requires continuous maintenance. Any software application or product developed will go through the maintenance phase. Software developers and programmers perform regular patches and updates as needed in the maintenance phase to address the changes in the company's or client's requirements. Throughout the maintenance phase, developers address different issues and try to add or remove certain functionalities in the project. Software ripples occur when one change impacts at least one other area of a software system after a change in one area. Below are the few real problems.

- Huge manual effort
- High probability of missing the files due to manual intervention
- Additional efforts by the developers
- Overall reduction of resolved change Requests.
- Impact to Customer Business value

The CIA tool will help in addressing the client's requirements. It helps the developer at the first level

to have an idea of how many changes are expected to be made for the given change request.

Unknown effects were found to be the main barrier to making new changes in a recent survey. [1].

As the file system gets bigger, it gets harder to manually identify changes' effects. Thus, increasing the significance of the Change Impact Analysis. CIAT is a way to predict the possible areas of effect. For example, the affected files and elements will be listed. The proposed CIA tool (StaticPy) will help to identify the affected files, methods, and lines of code in the file system in most applications.

The proposed tool (StaticPy) works for python, C++, Java and C applications.

This paper is presented as follows: Section-2 presents the related work; Section-3 describes about the proposed work, algorithm and the evaluation metrics. Section-4 shows the experimental results, Section-5 concludes the proposed work and Section-6 proposes the future work.

2. RELATED WORK

The following are the Change Impact Analysis methods already existing.

2.1 Following the edges of the system dependence graph (SDG) is the method used in CIA. Nodes representing actual and abstract program elements are found in an SDG, while edges are used to encode control and data dependencies [18]. G (f) is a directed graph of a function f. It is modeled as a directed graph. Class members, variables, constants, and other data components are represented by the nodes. The edges show how various data components are related to one another. Configuration-awareness requires representing the program's variability in the SDG. It made use of a conditional system dependence graph (CSDG), a variation of the SDG that shows variability as presence conditions. The relationships between classes form a network graph.

The system dependency graph extends earlier dependency representations by including collections of procedures. Data dependencies are relationships between program statements that define data [21]. If a statement in a program returns a value that is used, directly or indirectly, by another statement, it is data dependence. Data dependency graphs are used to represent these dependencies.

Data flow analysis provides dependency information about the data movement in the software system. It made use of a conditional system dependence graph (CSDG), a variation of the SDG that shows variability as presence conditions.

Consider a C program that has four changes as depicted in Figure. 1

C_0	C_1	C_2	C_3

void a() {
b();
c();
d();
}
	void a() {
	b();
	c();
	d();
	}
		void a() {
		b();
		c();
		d();
		}
			void a() {
			b();
			c();
			d();
			}

void b() {
e();
}
	void b() {
	e();
	}
		void b() {
		e();
		}
			void b() {
			e();
			}

void c() {
var2=1;
}
	void c() {
	var2=1;
	}
		void c() {
		var2=1;
		}
			void c() {
			var1=5;
			}

int d() {
return 0;
}
	int d() {
	exit(1);
	}
		int d() {
		exit(1);
		}
			int d() {
			exit(1);
			}

int e() {
f();
}
	int e() {
	f();
	}
		int e() {
		return 0;
		}
			int e() {
			return 0;
			}

int f() {
var1=0;
}
	int f() {
	var1=0;
	}		

Figure 1: Depicting changes in a C program.

The change history for the functions in the C file is shown in Figure. 2. It shows what kinds of changes were made to the file: addition, deletion, or modification. The dependence graph of function a() after C3 change is represented in Figure. 3

	C_0	C_1	C_2	C_3
a	**A**			
b	A			
c	A			M
d	A	M		
e	A		M	
f	A		D	

Figure 2: Depiction of the change history for the functions of the example system. The rows correspond to the functions and the columns to the changes, A,M,D are, respectively, Added, Modified and Deleted. Exit is not included because it is a C library function, external to the system being maintained.

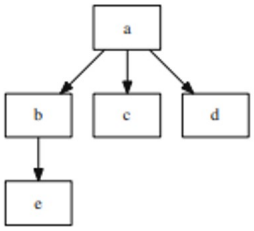

Figure 3: Dependence graph of a() immediately after change C3.

The System Dependence graph is also called Change Impact Graph (CIG) as it represents the change impact on other functions. Each node in the CIG can be of any three types mentioned below. Firstly, Unaffected Node No changes have been made to this function, nor have the functions it calls [23]. A Changed type affects other functions in the source code as a result of the changes made. Lastly, the Affected type is one whose source code has not changed, but at least one of its methods has.

2.2 By identifying common modelling and analysis techniques, the CIA expanded the Karlsruhe Architectural Maintainability Prediction (KAMP) approach for architecture-based study of change impact in many domains. Using a SAMM model instance, the KAMP Maintainability tool estimates cost and change effort for a change request [16]. SAMM is an open framework for forming and implementing software security strategies. To achieve software goals software architecture plays an important role. It estimates re-implementation costs along with re-deployment and upgrade costs for a given architecture. By using explicit architecture models, it evaluates maintainability for concrete change requests [17].

Additionally, it provides guidance in investigating estimation supports such as code and design properties, team organization, development environment, etc. As a second step in calculating change efforts based on semi-automated work plans and bottom-up effort estimation. As a software management and development tool KAMP is advantageous. Software management like re implementation, updating, installation is taken into consideration. The first phase is the top-down phase, in which the change requests are categorized into several change tasks and then the bottom-up phase, in which the effort involved in performing each change task is estimated. This approach demonstrates how change effort estimation may be integrated into a technique for predicting architectural maintainability, and it forecasts maintainability using change requests. The Maintainability Analysis Process has three different phases as shown below in figure 4[16]. As part of the preparation phase, a software architect sets up a description of software architecture for each architectural alternative. An Architecture Model is created for each Architecture Alternative.

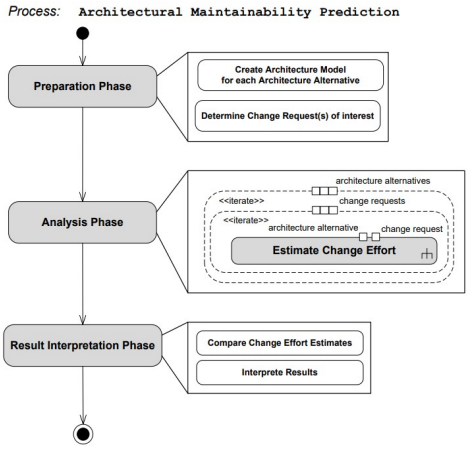

Figure 4: Architectural maintainability prediction.

The second step of the preparation process is to describe the proposed change requests. There should be a name, an explanation of the change cause and a list of existing architecture elements affected by the proposed change in the description. Following this phase is the maintainability phase, which involves estimating change effort for every change request and every architecture alternative. Interpreting the results, comparing calculated change effort and enriching them are all done in the Result Interpretation phase. Costs of software re-development as well as the costs of software management are considered for estimation.

2.3 Another model of CIA is based on Static Program Slicing for Industrial Software Systems. The calculation of static impact takes into account the static data gathered at compile time [19]. As a result, the computed set will be bigger than the calculated set using run-time data. It is a valuable tool for evaluating the impact of newly committed changes on a program using static program slicing. Without executing a program, static program analysis techniques gather information about the structure, as well as the run-time behavior of a program.

A program slice represents a specific behavior within a program by reducing the program to a set of statements (the slice) [7]. With this technique, only a portion of the program must be changed to generate a specific program behavior. The goal is to reduce the effort required to understand and maintain the program, by only considering a part of it. The quality and availability of regression tests are not constraints on static slicing. Different types of Programs Slicing techniques are: static slicing, dynamic slicing, simultaneous dynamic slicing, Quasi static slicing, and Amorphous slicing [22].

2.3.1 Static slicing

In terms of program reachability, static slicing can be expressed using a program dependency graph. Using the slice, all PDG verticals from which the given criterion can be reached are considered. Starting at the slicing criterion, a backward traversal of the program's control flow graph (CFG) or PDG is performed to gather statements and control predicates. Forward slices represent the set of statements that depend on the slicing criterion, with each statement being "dependent" on it. Using this tool, we can predict what changes will be brought about by the slicing criteria in that program.

2.3.2 Dynamic Slicing

Statements in static slices do not influence the values of variables in the execution. A program's execution may be affected by the value inputted. When the program is running, the input it received is inspected, but just the statement that caused the failure is looked at. Variables that may be affected by statements can be found using dynamic analysis. It benefits from managing arrays and pointers in real time. Dynamic slicing treats each array element independently, whereas static slicing treats each array element as an array as a whole.

2.3.3 Simultaneous Dynamic Slicing

A New slicing technique combines test case slicing with program slicing. In simultaneous dynamic program slicing, a set of test cases is applied simultaneously to the dynamic slicing technique, yielding executable slices valid only for a single parameter [22].

2.3.4 Quasi static slicing

A hybrid slicing technique is a combination of static and dynamic slicing. It is called Quasi static slicing. While static slicing analyzes the code during compile time without knowing the input variables, dynamic slicing analyzes the code at run time. Practically, quasi slicing analyzes the code without knowing the input variables.

2.3.5 Amorphous Slicing

By syntax-preserving slicing techniques, the program statements are discarded based on the slicing criteria and the syntax remains unchanged despite slicing. By contrast, amorphous slicing involves any program statement.

In comparison to all other slicing techniques the slices formed are very small.

There are two main types of program slicing: static program slicing or dynamic program slicing, which removes statements that have no effect on a certain point of interest. There are two fundamental static analysis-based program slicing approaches. Backward slicing was proposed by Weiser. Control-flow graphs (CFGs) are used for backward slicing analysis, which identifies areas of a program that contain bugs to assist developers. As a result of backward-slicing developers can locate the parts of their program that contain bugs by analyzing control-flow graphs (CFGs).

Similarly forward slicing was proposed by Horwitz et al. Instead of slicing a CFG, they slice its dependency graphs. Executable slices are syntactically correct slices that produce executable programs that exhibit a portion of the original program's behavior. All statements and language constructs in P that may affect the values of variables in V at location L are considered backward slices of P. Figures, for example, represent the location of the program (L, V) as a forward slice of P [20]. The backward slice in relation to the slicing criterion is shown below (3, z).In debugging, backward slicing can identify statements that cause a particular error state, while forward slicing can identify the code that may change.

1. $x := 6$
2. $y := 5$
3. $z := y - 1$
4. $r := x + z$
5. $z := x + y$

Figure 5: Original Program

1.
2. $y := 5$
3. $z := y - 1$
4.
5.

Figure 6: Backward slicing on (3,z)

1.
2.
3. $z := y - 1$
4. $r := x + z$
5.

Figure 7: Forward slicing on (3,z)

The following is the list of existing Change Impact Analysis tools.

Technique	Tool support			
	Name	Input	Output	Language
Briand et al. [27]	*Columbus*	Object-oriented system	Structural coupling measures	C++
Zimmermann et al. [55]	*ROSE*	Software historical repositories; current program; changes	Impacted parts	Java
Apiwattanapong et al. [30]	*JAT*	Execution information; proposed changed methods	Impacted methods	Java
Ramanathan et al. [57]	*Sieve*	Programme binaries of original and modified program	Impacted methods and code regions in modified program	C
Canfora et al. [59]	*Jimpa*	A change request description; historical source files repositories	Impacted files	Any*
Huang et al. [60]	*JDA*	Changes; the program; some executions	Impacted methods and fields	Java
Haftori et al. [38]	*Impala*	The system; changes	Impacted elements	Java
Prabhuswyk et al. [58]	*JRC*^{2}*M*	A project	Conceptual coupling measures	Any*
Petrenko et al. [65]	*JRipples*	The system; changed classes	Impacted classes	Java
Gethers et al. [70]	*LDA*	A software project	Relational topic based coupling	Any*

*Any, represents that this tool supports the CIA technique, which is immune to its supported language type.

Figure 8: Tool support for change impact analysis technique

There are some tools that operate on the version history and few tools which work only for a particular language.
For example, ROSE is a tool that operates on the version history, and is able to detect coupling between items based on the version history. After an initial change, the ROSE tool can predict the location which needs to be changed. But if we need to make any code changes which were not addressed previously, it is difficult for the tool to predict the accurate results. The accuracy of ROSE is more than 70% [26]. StaticPy does not require any version history to perform impact analysis. It will work for any type of change.

3. PROPOSED METHODOLOTY

Problem Statement: To develop an automated CIA Tool (StaticPy) that helps to identify the impacted files, methods, and fields that are affected due to the proposed changes.

In this section will discuss the proposed work, proposed algorithm and the evaluation metrics used.

3.1. Proposed Work

There are three ways to do CIAT. They are as follows:
(1) Regular Expressions
(2) Tokenizer
(3) Abstract Syntax Tree (AST)

In this proposed work, Tokenizer is chosen among the three because of the following reasons:
i) Regular expressions can be difficult to write correctly and maintain for different language files because each language has different syntax and semantics.
ii) In Regular expressions, it is next to impossible to detect edge cases in all circumstances.
iii) In AST, it only works for Python, and it also requires syntactically valid Python (with a few minor exceptions).
iv) Tokenizer can work with incomplete or invalid Python.
v) Using Tokenizer edge cases can be avoided.

Thus, in this paper tokenizer is chosen for its benefits and usability for all the languages.

3.2. Proposed Algorithm

Input: change ticket, GitHub repository link of the application
Output Objectives:
i) Affected file names
ii) Affected elements in the files
iii) Line numbers of affected lines

Journal of Theoretical and Applied Information Technology
15th April 2023. Vol.101. No 7

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ALGORITHM

Input:

TokenList1: List of tokens generated from restructured data using the Treebank tokenizer function.

TokenList2: List of keywords extracted from the Yake tool from a ticket raised by the user.

Output:

ResultArray: list of **affected** elements from the given ticket.

Algorithm:

```
 LineNumber: =1
D[ ] empty dictionary
ResultArray[]
ForEach word :  File.Line do:
      If (word==’\n’ ) do:
          LineNumber: +=1
      Else If word in TokenList1 do:
        D[word]:=D.get(word,[]).add(LineNumber)
      End If
End For
ForEach keyword: TokenList2 do:
         Temp: =D.get(keyword)
          ResultArray.add (Temp)
End For
Display (ResultArray)
```

The following diagram shows the overview of the proposed methodology.

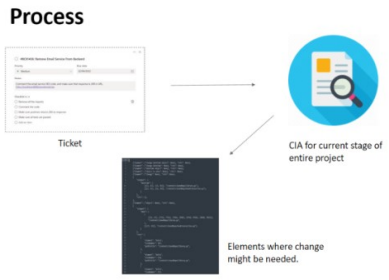

Figure 9: Overview of proposed methodology

1. To get started with the tool, you need to install all the required packages and modules.

2. Then the repository is cloned from the given GitHub link.

3. Then restructure the cloned files by changing the *self*, this, etc. with their respective class name. For this go through the file and find all the places where the class keyword exists. Then the algorithm checks the text to the right of it and extracts the class name. Then in the code following the location where the class keyword is found i.e., in the entirety of that class, we replace the *self* (for python), this (for C++, java) etc. with the extracted class name.

4. In order to prevent the CIA from evaluating the remarks, the data has now been reorganized by eliminating them. This step must be done without modifying the total length of the code. For this each file is checked for comments with the help of regex. As there are 2 types of comments in all the major languages. They are: Single line comments and multi-line comments. Both these comments are identified and replaced with an unused symbol without altering the number of lines in the file.

5. Then restructure the data by removing the unwanted symbols, which may cause unwanted results while creating the data structure. For this, a regex is used to find these symbols and replace them with an identifiable and unused token.

6. Now tokenize these files and find the spans of each token. For this Treebank tokenizer is used. The StaticPy tool uses regular expressions to tokenize the given content in the file. It assumes that the text has already been split into sentences.

i. It treats punctuation as a separate token.

ii. It splits at a full stop at the end of sentence.

iii. When there are commas between quotes, spaces must come after them.

Tree Bank Tokenizer is imported into the project from nltk module of Python.

```
>>> from nltk.tokenize import TreebankWordTokenizer
>>> s = '''Good muffins cost $3.88\nin New York.  Please buy me\ntwo of them.\nThanks.'''
>>> TreebankWordTokenizer().tokenize(s)
['Good', 'muffins', 'cost', '$', '3.88', 'in', 'New', 'York.', 'Please', 'buy', 'me', 'two'
>>> s = "They'll save and invest more."
>>> TreebankWordTokenizer().tokenize(s)
['They', "'ll", 'save', 'and', 'invest', 'more', '.']
>>> s = "hi, my name can't hello,"
>>> TreebankWordTokenizer().tokenize(s)
['hi', ',', 'my', 'name', 'ca', "n't", 'hello', ',']
```

Figure 10: TreeBank Tokenizer Examples

7. The number lines for these tokens are calculated. For this, the number of new line characters in the file before the token is calculated. With the help of this, you can get the line number of the token.

8. Then the algorithm starts filling the data structure which is a dictionary with key value pairs. The keys are tokens, and values are their spans in the entire file. Here in a repository, different files can have the same token, so the algorithm is designed to also add the filename along with the span to individually identify them.

9. Next find all the related elements of these tokens. For this go to the span location by traversing through the dictionary then check if that's a valid location and then check for its references, assignments, etc. which all get affected based on the change of this token and add them as values to the token in the data structure along with their line number and file name.

10. Then the algorithm takes the change ticket, for the ticket given assuming the person issuing the ticket has basic knowledge of the code base. For example, these tickets can be considered as Jira tickets. Now, we need to extract keywords from the ticket. For keyword extraction yake keyword extractor is used.

3.2.1. YAKE

It stands for yet another keyword extractor. It is an unsupervised automatic keyword extraction algorithm that is very efficient at extracting meaningful keywords in a corpus. It works by giving probability to each word, which depicts how meaningful the word is. The lower the probability, the better the chances that the word is meaningful. It doesn't depend on corpus nor does it need to be trained on a particular set of documents. It follows a 5-step process [2] which is depicted below:

Figure 11: Process of Yake

11. Then, using global data structure, perform a breadth-first search to find all references to keywords in all the project's files and store the results in a result array.

12. Step-11 is repeated for all the extracted keywords and updates the result array.

13. Finally, the affected elements from the result array are displayed.

3.3 Evaluation Metrics

The following is the confusion matrix [1, 3] used to calculate the Evaluation metrics- Accuracy, Precision and Recall.

Actual Class	Predicted Class	
	Negative	Positive
Negative	TN (True Negatives)	FP (False Positives)
Positive	FN (False Negatives)	TP (True Positives)

Figure 12: Confusion Matrix

A measurement, calculation or specification is accurate if the result conforms to the standard or value required.

$$Accuracy = \frac{truepositives + truenegatives}{totalexamples}$$

A precision can be defined as the percentage of predicted positive cases that are correctly classified as positives

$$Precision = \frac{truepositives}{truepositives + falsepostives}$$

This is a measure of what proportion of actual positive cases are correctly classified as positive.

$$Recall = \frac{truepositives}{truepositives + falsenegatives}$$

4. RESULTS AND DISCUSSIONS

The values TP, TN, FP, FN are calculated as follows:

TP (True Positive): If the tool predicts a line number to be affected which is actually been affected when validated manually.

TN (True Negative): Consider a case as TN when a line number predicted by a tool is not affected and the line number is actually not affected when verified manually.

FP (False Positive): Consider a case as FP when a line number predicted by the tool is affected, and the line number is actually not affected when verified manually.

FN (False Negative): Consider a case as FN when a line number predicted by the tool is not affected, and the line number is actually affected when verified manually.

i) Number guessing game written in Python

ii) Library Management System written in Python.

iii) A Book Reader App written in C++.

iv) Text Based Adventure Game written in Java.

While testing the StaticPy tool, the results were calculated using the tool and verified with the manually calculated results.

The following is an example (snippet) showing how the Accuracy is calculated in the number guess game for tickets having the data "Change lower."

Using these standards as metrics, the results for the following projects are calculated:

```
1 from random import randint (TN)
2 import math (TN)
3
4 lower = 3 (TP)
5 upper = 20 (TN)
6 limit = 10 (TN)
7
8 x = randint(lower, upper) (FP)
9 print('\n\tYou\'ve only ', (TN)
10        limit, (TN)
11       ' chances to guess the integer!\n') (TN)
12
13 # for calculation of minimum number of
14 # guesses depends upon range
15 while limit != 0: (TN)
16     limit = limit - 1 (TN)
17     # taking guessing number as input
18     guess = int(input('Guess a number:- ')) (TN)
19
20     # Condition testing
21     if x == guess: (TN)
22         print('Congratulations you did it in ', (TN)
23               count, ' try') (TN)
24         # Once guessed, loop will break
25         break (TN)
26     elif x > guess: (TN)
27         print('You guessed too small!') (TN)
28     elif x < guess: (TN)
29         print('You Guessed too high!') (TN)
```

Figure 13: Application – Code snippet

The tool predicted 1 TP, 27 TN, 1 FP, and 0 FN values, as seen in the image above.

In the above Code Snippet, shown in figure. 13,

- Due to the absence of information on the 'lower' attribute, Line-1 is TN.
- Line-4 refers to TP as we have information about the 'lower' attribute.
- Line-8 is FP since it contains information on lower attributes, but it's misleading because we are trying to change the value of only the lower attribute.
- Other than that, all lines are TN, since there is no information about 'lower attribute' in them.

The output given by the developed tool for input ticket “change lower” is as follows:

```
/content/clonedRepo/number_guess.py
[
   {"element": {"change lower": None}, "refs": None},
   {"element": {"change": None}, "refs": None},
   {
      "element": {"lower": [([4, 8], "/content/clonedRepo/number_guess.py")]},
      "refs": [
         {
            "element": "x",
            "lineNumber": 8,
            "pathToFile": "/content/clonedRepo/number_guess.py",
         }
      ],
   },
]
```

Figure 14: Output of Developed CIA Tool for give code snippet depicting elements, files and lines affected.

Various tickets are tested for each of the above-mentioned four applications to evaluate their metrics.

Fig. 15 The image below shows the True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN) and the Accuracy for different tickets raised on the Number Guess Game application.

	TP	TN	FP	FN	ACCURACY
Change limit	3	25	1	0	0.965
Change lower	1	27	1	0	0.965
Change upper	1	27	1	0	0.965
Change randint	2	27	0	0	1

Figure 15: Metrics for number guess game tickets.

The below graph Fig. 16 shows the Accuracy of the tickets mentioned in Fig. 15. The average accuracy is 97.3%.

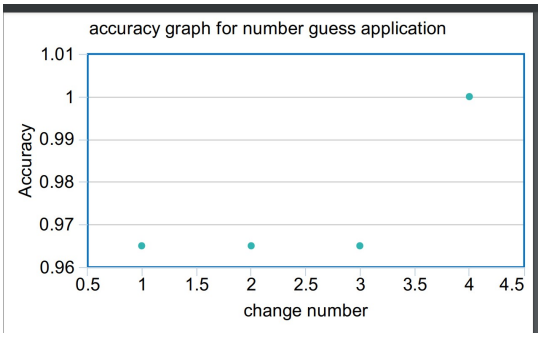

Figure 16: Accuracy graph for number guess application.

The True Positive (TP), False Positive (FP), True Negative (TN), False Negative (FN) and the Accuracy of different tickets are calculated for the Book Reader application.

The below figure shows the metrics calculated for the Book Reader application.

TICKET/REQ	TP	TN	FP	FN	ACCURACY
Change the setter's name	1	666	4	0	0.994
Update GetBook Method in BookReadingSession Class	1	657	13	0	0.98
Update Next Page Method BookingReadingSession Class	1	658	12	0	0.9806
Change var.pages and recheck code	3	666	2	0	0.997

Figure 17: Metrics for book reader application.

The below graph shows the accuracy of the Book Reader Application for the tickets mentioned in Fig. 17.

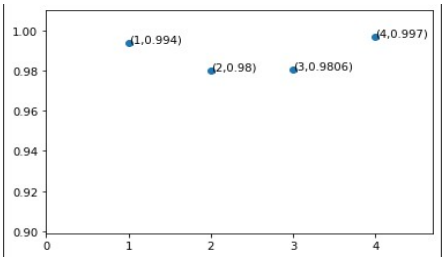

Figure 18: Accuracy graph for book reader application.

The average accuracy for Book Reader Application is 98.7%.

The accuracy calculated for the Library Management Application using the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) for various tickets is shown in the figure below (Fig. 19).

TICKET/ REQ	TP	TN	FP	FN	ACCURACY
Increase fine by 50%	1	380	7	0	0.97
Change the username	3	940	6	0	0.99
Increase interest counter by 2	1	384	3	0	0.992
Change value of day as previous day	2	379	7	0	0.98

Figure 19: Metrics for library management application.

The accuracy graph for the Library Management application is shown in the below graph Fig. 20.

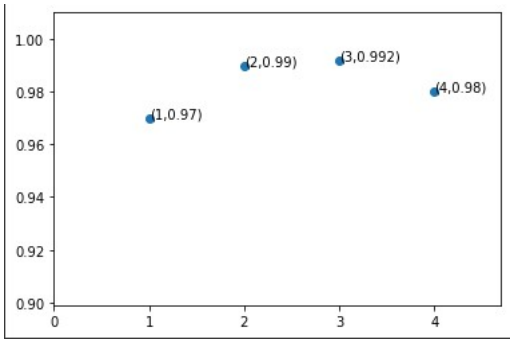

Figure 20: Accuracy graph for library management application.

The average accuracy for Library Management Application is 98.3%.

The below figure Fig. 21 shows the accuracy calculations of the java application which is Text based Adventure game application.

Ticket/REQ	TP	TN	FP	FN	ACCURACY
Initialize position with 100	2	172	3	0	0.983
Initialize the maxHp to a random value	2	171	4	0	0.971
Change value of hpzomb from 7 to 10	1	172	4	0	0.977
Change the atk and defence base values to 10	4	165	8	0	0.954

Figure 21: Metrics for text-based adventure game application

The accuracy of different tickets raised in Java application is plotted in the following graph.

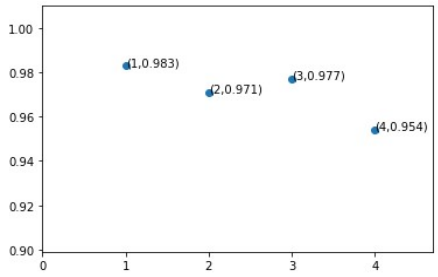

Figure 22: Accuracy graph for text based adventure game application

The average accuracy for Text Based Adventure Game Application is 97.1%.

We determined the accuracy of the developed CIA tool (StaticPy) by averaging all four applications, accuracy values that were tested with the developed tool. The above four were the best applications that produced the best accuracy numbers.

The average accuracy of the above applications tested using developed StaticPy CIA Tool is 97.8%.

5. CONCLUSION

This paper is primarily focused on developing an automated change impact analysis tool (StaticPy). The tool was developed and tested using four different software applications in different languages. The average accuracy of the tool is 97.8%. StaticPy uses change request (ticket) and the application repository as input to determine the impact of the changes on the application. With the help of the proposed tool the developer will have an idea about where the changes are expected to be made, the amount of work to be done for a change request and plan the work accordingly.

The proposed tool uses Static analysis - tokenizer concept, and intermediate data structure for change impact analysis. ROSE and ImpactMiner use static analysis and dynamic analysis techniques to determine the impact of code changes on the system [26-27]. Both the tools apply data mining to version histories to predict Change impact Analysis.

The proposed work does static analysis, i.e., assuming the project is complete and up and running. StaticPy is implemented in python3. It takes the GitHub repository links (the application link) and change request (ticket) as input. Then it gives all the affected elements, file names and line numbers as the output. The accuracy may change from application to application.

Accuracy was determined by comparing the results produced by the CIAT with those obtained through manual testing. The developed tool has been tested on Python, C++ and Java applications. Comparing the developed tool with existing tools, the tool was tested against applications that other papers have tested. Number guess and a library management system were the applications. The tool was tested on the same applications and it produced better results than the previous ones which had an average accuracy of 93% and the proposed tool

(StaticPy) average accuracy is 97.8%. The proposed tool (StaticPy) can be used to find the change impact analysis of different languages, for example- python, C++, Java.

6. FUTURE SCOPE

The CIA Tool developed can be improved by improving the keyword extraction algorithm. During the formation of the data structure, using improved tokenizer and spans can improve accuracy. In the future if there are better algorithms, they can help improve accuracy even more.

Further if there are ways to traverse a GitHub repository without cloning, then that helps to decrease the storage space required. Right now, there is no support for scanning specific parts of code or specific files against the ticket, which could also provide that functionality and improve the features of the tool.

CONFLICTS OF INTEREST

The author(s) declare that there is no conflict of interest regarding the publication of this paper.

FUNDING STATEMENT

We thank Vasavi College of Engineering (Autonomous), Hyderabad for the support extended towards this work.

ACKNOWLEDGMENTS

The paper "Conceptualization, investigation, methodology, formal analysis, Identification of software for experimentation, resources, implementation, validation, writing, original draft preparation, editing and visualization of the result have been done by the first author. Reviewing the work and supervision is done by the second author.

REFERENCES:

[1] Sufyan Basri, Nazri Kama, Roslina Ibrahim and Saiful Adli Ismail, "A Change Impact Analysis Tool for Software Development Phase", Internal Journal of Software Engineering and its Applications, Vol. 9 No. 9 (2015).

[2] Maria Kretsoua, Elvira-Maria Arvanitou, Apostolos Ampatzoglou, Ignatios Deligiannis, Vassilis C.Gerogiannis, "Change impact analysis: A systematic mapping study", Journal of Systems and Software, ScienceDirect, Volume 174, April 2021, 110892.

[3] T. Jalaja, T. Adilakshmi, "Change Impact Analysis using Python for Java Applications", International Journal of Recent Technology and Engineering (IJRTE), ISSN: 2277- 3878 (Online), Volume-9 Issue-1, May 2020.

[4] Haipeng Cai, Raul Santelices and siyuan Jiang,"Prioritizing Change-Impact Analysis via Semantic Program Dependence Quantification", IEEE Transactions on Reliability,Vol.65, No.3,September 2016.

[5] T. Jalaja, T. Adilakshmi," Automation of Impact Analysis", International Journal of New Innovations n Engineering & Technology, Vol.11 Issue4, September 2019.

[6] Sufyan Basri, Nazri Kama, Roslina Ibrahim, Saifuladli Ismail, " A Change Impact Analysis Tool for Software Development Phase", International Journal of Software Engineering and its Applications 9(9):245-256, September 2015.

[7] Mithun Acharya, Brian Robinson, "Practical change impact analysis based on static program slicing for industrial software systems", Proceedings of the 33rd Internal conference on Software Engineering, pages 746-755, May 2011.

[8] Tie Feng, J.I. Maletic, "Applying Dynamic Change Impact Analysis in Component-based Architecture Design", Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06), IEEE, 2006.

[9] https://ieeexplore.ieee.org/document/6032516

[10] Nazri Kama, Faizul Azli, " A Change Impact Analysis Approach for the Software Development Phase", 19th Asia Pacific Software Engineering Conference, IEEE, 2012.

[11] Hassan Osman Ali, Mohd Zaidi Abd Rozan, Abdullahi Mohamud Sharif, "Identifying challenges of change impact analysis for software projects", International Conference on Innovation Management and Technology Research, IEEE May 2012.

[12] Hoa Khanh Dam, Aditya Ghose, "Automated change impact analysis for agent systems", 27th IEEE International Conference on Software Maintenance (ICSM), Sep 2011.

[13] Daniel Amyot, Nikolai Mansurov, Gunter Mussbacher, "Understanding Existing Software with Use Case Map Scenarios", Conference: Telecommunications and beyond: The Broader Applicability of SDL and MSC, Third

International Workshop, SAM 2002, Aberystwyth, UK, June 24-26, 2002.

[14] Michelle L. Lee "Change Impact Analysis of Object Oriented Software. Technical Report ISE-TR-99-06" George Mason University Fairfax Virginia Dec 1998.

[15] Heiko Koziolek, Bastian Schlich, Steffen Becker, Michael Hauck "Performance and Reliability Prediction for Evolving Service-Oriented Software Systems Industrial Expert Report" August 2012.

[16] Johannes Stammel, Ralf H. Reussner "KAMP: Karlsruhe Architectural Maintainability Prediction" Forschungs zentrum Informatik (FZI) 15 May 2014.

[17] Birgit Vogel-Heuser, Robert Heinrich, Suhyun Cha, Felix Ocker, Sandro Koch "Maintenance effort estimation with KAMP4aPS for cross-disciplinary automated PLC-based Production Systems - a collaborative approach" Vol 50 issue July 2017 Elsevier.

[18] Florian Angerer , "Change impact analysis for maintenance and evolution of variable software systems" December 2017.

[19] http://www0.cs.ucl.ac.uk/staff/mharman/sf.html

[20] Ekincan Ufuktepe, "A Program Slicing-based Bayesian Network Model for Change Impact Analysis Evolution styles: Foundations and models for software architecture evolution" Barnes et al., 2014J.M. Barnes, D. Garlan, B. Schmerl.

[21] AllenR., Kennedy, K.: "Optimizing Compilers for Modern Architectures: A Dependence-Based Approach." Morgan Kaufmann, Burlington (2001).

[22] Sasirekha Neelamegam, Edwin Robert "Program Slicing Techniques And Its Applications", International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.3, July 2011.

[23] Daniel M German , Gregorio Robles, Ahmed E. Hassan "Change impact graphs: Determining the impact of prior code changes", IEEE International working conference on Source Code Analysis and Manipulation, 28-29 September 2008 Beijing China.

[24] Hasan Alkaf, Jame leddine Hassine "An automated change impact analysis approach for User Requirements Notation models" Elsevier, November 2019.

[25] Barnes et al, Barnes, D. Garlan, B. Schmerl , "Evolution styles: Foundations and models for software architecture evolution",2014J.M

[26] Thomas Zimmermann, Student Member, IEEE, Peter Weißgerber, Stephan Diehl, and Andreas Zeller, Member, IEEE Computer Society "Mining Version Histories to Guide Software Changes", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 31, NO. 6, JUNE 2005, Pgno-429 to 445.

[27] B. Dit et al., "ImpactMiner: A tool for change impact analysis," in Companion Proc. 36th Int. Conf. Softw. Eng., 2014, pp. 540–543.