Software Change Impact Analysis for Sensitive Applications

A MAJOR PROJECT-II THESIS REPORT

SUBMITTED IN PARTIAL FULFILMENT OF REQUIREMENT
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
SOFTWARE ENGINEERING

Submitted By – **Sumit Bansal**
(Roll No. 2K18/SWE/24)
Under the supervision of **Dr. Ruchika Malhotra (Associate Professor)**
Delhi Technological University

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering) Bawana Road, Delhi-110042

June, 2020

DECLARATION

I hereby declare that the Major Project-II work entitled "**Software Change Impact Analysis For Sensitive Applications**" which is being submitted to Delhi Technological University, in partial fulfillment of requirements for the award of the degree of Master of Technology (Computer Science and Engineering) is a bona fide report of Major Project-II carried out by me. I have not submitted the matter embodied in this dissertation for the award of any other degree or diploma

Place: Delhi

Sumit Bansal **2K18/SWE/24**

CERTIFICATE

This is to certify that the Major II Report entitled "**Software Change Impact Analysis for Sensitive Applications"** submitted by **Sumit Bansal** (2K18/SWE/24) for the partial fulfilment of the requirement for the award of degree of Master of Technology (Software Engineering) is a record of the original work carried out by him under my supervision.

Place: Delhi

Date: **Project Guide Dr. Ruchika Malhotra Associate Professor**

Department of Computer Science & Engineering

Delhi Technological University

ACKNOWLEDGEMENT

First of all, I would like to express my deep sense of respect and gratitude to my project supervisor **Dr. Ruchika Malhotra** for providing the opportunity of carrying out this project and being the guiding force behind this work. I am deeply indebted to him for the support, advice and encouragement he provided without which the project could not have been a success.

Secondly, I am grateful to Dr. Rajni Jindal, HOD, Computer Science & Engineering Department, DTU for her immense support. I would also like to acknowledge Delhi Technological University library and staff for providing the right academic resources and environment for this work to be carried out.

Last but not the least I would like to express sincere gratitude to my parents and friends for constantly encouraging me during the completion of work.

Sumit Bansal

Roll No – 2K18/SWE/24

M. Tech (Software Engineering)

Delhi Technological University

ABSTRACT

With the software intensive systems growing at a rapid rate, the system safety controls have become completely software defined. Therefore, the impact analysis of Safety - involved systems become necessary, to take care of any software changes during the maintenance phases of such systems. However, there hasn't been much work done in this scope so far. So, through this thesis we propose a new tool to choose the appropriate approach for the software change impact analysis along with explaining the software IA concepts, describing its importance and providing an insight into its current hierarchy and classification. We also present the survey results based on a research methodology to get the practitioner's view point of Change IA.

Keywords: SUT CIA, IA, Slicing, Program Analysis, Sensitive Application.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Abbreviations and Nomenclature

- IA – Impact Analysis
- CIA – Change Impact Analysis
- SDLC – Software Development Life Cycle

CHAPTER-1

Introduction

With a fast increase in computer based technologies, the software intensive systems continue to evolve across a myriad of industries. Due to this rapid increase in the demand of software-operated systems, the system safety control (like braking systems, airbag systems etc.) has been completely taken up by the software. Safety-critical systems or Safety-involved systems address the applications in which any failure can be catastrophic; it may result in a heavy loss of life, property and environments. Therefore, it becomes necessary to come up with new methods for improvements in the non-functional requisites of life-critical systems. Moreover, for such safety-critical applications, it becomes essential to have a proper impact analysis. An improper impact analysis can cost up to billion-dollars, hence IA should be performed with great caution taking into account all the potential changes that one may have to make as a result of a change in a variable or a function.

Certainly we need more support for the systems and resources for more detection of errors. Also if the process improvement were something as easy as modifying developer preferences before agreeing to execute error management strategies, this might probably save hundreds of millions of pounds for several industries. For software and computer engineering, by measuring fault-proneness early in the software using reference model, we can attain a great quality in a cost-effective approach. The crucial thing is to have relevant information that can be used for concentrating on inspection and validation efforts mostly on affected parts, and to improve the efficiency of maintenance and testing. Moreover, it has become a challenging task to maintain large systems, particularly object-oriented software. Modification is indeed an essential property of all software, and is needed to keep the system consistent and free from any faults and it is unavoidable during maintenance. This provides an adequate approach to avoid severe collapse of the software when modifications are incorporated. Hence this becomes our purpose of research on this topic.

Impact analysis predicts modifications in any system for compliance with the change. There are various Impact analysis tools available for various different purposes, for instance change impact analyser, ImpactMiner, CodeSurfer, EAT, eROSE, Imp, Jripples, FaultTracer, Kaveri etc. some of these tools are made for the academic purpose ad are a result of the thesis research work of different people, some others are available as open source, such that the software developers can make use of it while contributing important changes, while others like JArchitect are commercially used.

Software change cannot be avoided during the maintenance phases, especially object oriented software. The impact analysis can also be used along with the fault proneness prediction to overcome the problem of the software failure likelihood due to the change made on OOS faulty class. This becomes necessary as the software development is surging through the globe, there is a need to maintain high quality software systems in order to avoid serious faults that can cause severe damage or failure.

1.1 Motivation

As the large scale technology advances, with an expectation to be a long term evolution, the software testing and analysis becomes non-negotiable. Every year new stories of software failure in the software development industry comes out, with consequences often being detrimental, causing a loss to both life and property. One of the examples of such an accident, is F-35 fighter plane software glitch that caused target detection problems, another such software glitch was reported in SolarCity Corp that resulted in an under-evolution of $400 million in its acquisition. In all these accidents inadequate change management is reported to be the most common cause. Therefore, it becomes necessary to understand how a software change can impact the over-all system. This software evolution of safety-involved systems can be performed using Impact analysis. IA has great advantages when performed properly during the development and maintenance of the software life time. However, there is still a lack proper research survey the field of impact analysis.

Several researchers already concluded that the process to conduct sensitivity analyses or IA during the life-cycle of software creation and support processes has great advantages. Several tools have been developed for educational researches that encompass the advantages of IA, but there is a notable shortage of such tools made commercially accessible to users.

Internally executing regression tests is an approach developers are using to guarantee their action doesn't break the design. Development team can decide execute all tests or to pick a sub-set of tests that are applicable to the change. A downside here is that it can take some time to execute tests locally, particularly when the designer wants to execute all tests.

It should be noted that there are plenty techniques for impact analysis that've been designed, developed, and tested but only a few of these few tools are found to provide such techniques to enterprise developers. Such methods are said to enhance the maintenance of the application and lower the number of errors that are being incorporated into the primary framework. It logically follows that all these techniques can embody in functional tools for the software developers to use them. Moreover, there is an accepted need for the implementation of an appropriate impact analysis tool to facilitate systematic and continuous use of approaches for impact assessment.

Thus, with this paper we aim to survey various error prevention techniques based on impact analysis tools available in the market both commercially and academically and then divide them in different categories based on some development structure, we then further focus on Change Impact Analysis technique and conduct a survey by interviewing fourteen practitioners of CIA and the results and findings of this survey should thus lay a basis to conduct future work on IA tools. CIA is used to evaluate the software of safety-critical systems and it is a labour intensive work. CIA is necessary to find how a change will propagate through a real project baseline. Furthermore, we propose a tool such that the user is able to select the impact analysis approach based on their specific requirements while keeping in mind the General Data Packet Regulation (GDPR) i.e. often termed as one of the biggest improvements ever made in the history of data protection. We'll address the use impact analysis tool to analyse the change in a company's system as a result of the GDPR compliance.

1.2 Scope

With so many tools available to perform IA, we keep our focus limited to only those tool that helps the developer to analyse and find the potential consequences of a change to a system, before the changes have been implemented or integrated into the actual project standard.

It should be marked that there are various tools available to help in the maintenance phases of a software, that are out the scope of IA, such as FindBugs, Lint, MALPAS etc., are based on static code analysis, where the software is analysed without executing the program code. Other tools available for software maintenance analysis are dynamic program analysis, in which the software is analysed by executing programs in virtual or real processors. Both of these tools should be used with the impact analysis tool in order to increase the over-all efficiency of the analysis and get the best results.

The concept "impact analysis" is recognized to be a broad term. IA can be carried out on several levels, from a code level to a part of the framework, and on a logical level. Several tools that cover these variants of analysis have been identified, however a lot of them were considered out of scope for this paper's purpose. Numerous tools have been established that encompass these variations of analysis but many of them were regarded outside the scope for the purpose of this paper.

The thesis foresees:

--Offer software impact analysis concepts. Describe how such an analysis of the impact of change is essential, and provide an insight into current classifications and solutions which have already been suggested before.

--Provide definitions of the European Union's General Data Protection Regulation and its historical perspective.

--Propose a tool from which decides an appropriate approach for software CIA taking into account different parameters.

--Attempts to examine a range of established tools available for IA, both educationally and commercially, then characterize both as per a framework.

-- Survey results based on a research methodology to get a practitioner’s view point of Change Impact Analysis.

CHAPTER-2

Background

Maintenance of software can be classified into four categories: adaptive, corrective, perfective and preventive. Most software developers end up loosing significant time and resources on maintenance of corrective as well as adaptive software, resulting in exaggerated expenses and production delays. Software maintenance can be either reactive or proactive, while the reactive maintenance type is performed after the fault has been discovered, the proactive maintenance focuses on saving time and resources. Perfective and preventive are examples of proactive maintenance while the other two i.e. adaptive and corrective are example of reactive software maintenance.

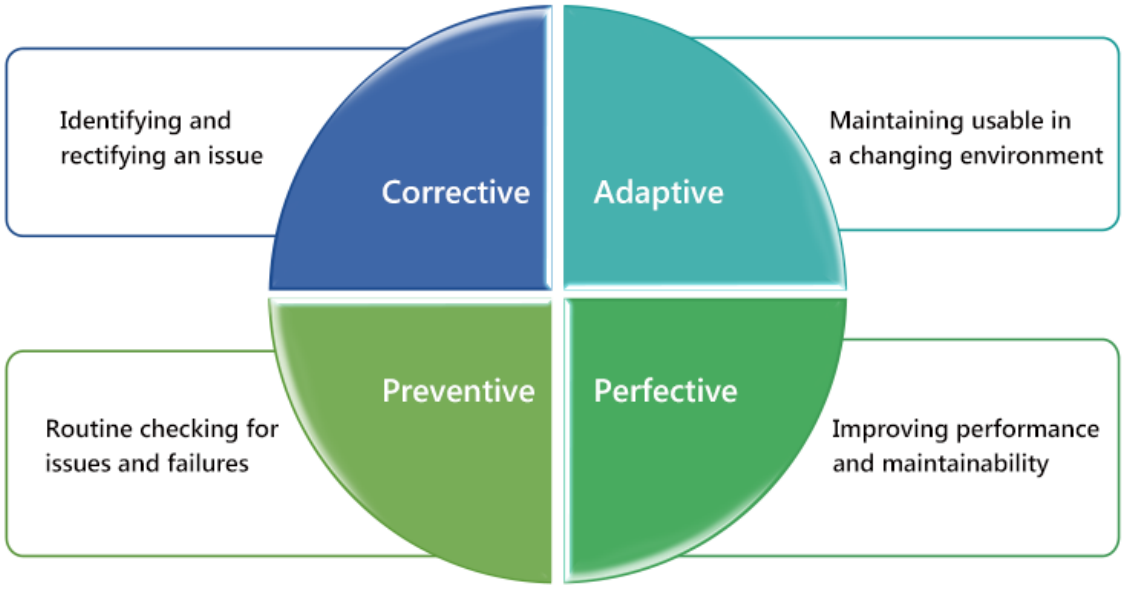


Figure 1 – Software maintenance types

In scope of this research, we are concerned with a proactive maintenance, called as preventive maintenance. Preventive maintenance is performed routinely on a device to reduce the probability of its failure. Preventive maintenance is practiced while the device is still operating fine, so as to prevent it from breaking down unexpectedly. It prevents any unanticipated downtime and high costs from accidental malfunct----------ion of the equipment. Until there is a serious issue it needs proper preparation and scheduling of repairs on devices, and maintaining clear records of previous inspections and service reports. The precise required preventive maintenance can differ considerably on service and type of device. American National Standards Institute (ANSI) standards are used to assess potential type of inspections and maintenance necessary and how frequently to carry out them.

Preventive maintenance is much more than just doing routine check of the equipment or device in operation. Proper records of each inspection should be maintained for future references as this helps estimating when the change is anticipated to occur.

2.1 Preventive maintenance and Code Evolution

V'aclav[] dealt with the necessity of effective development of software tools. Many development of software occurs in a constant state of change. The IA and concept location are critical tasks involved in the successful advancement of a code. V'aclav states that for such fundamental task, continual improvisations are required in the tool support.

It is known that almost any bit of software has to be maintained eventually. Nevertheless, as the process of software development enhances, it is evident that development coincides with the evolution of code. The success factors included in maintenance of software should be used over the complete software lifespan.

Software development teams must consider guiding principles to establish better habits among their developers prior to actually committing code, to safeguard the baseline. This is much more engaged in practicing appropriate preventive maintenance and reliable code evolvement than simply making and committing changes.

The software developers should do the following before the code commit–

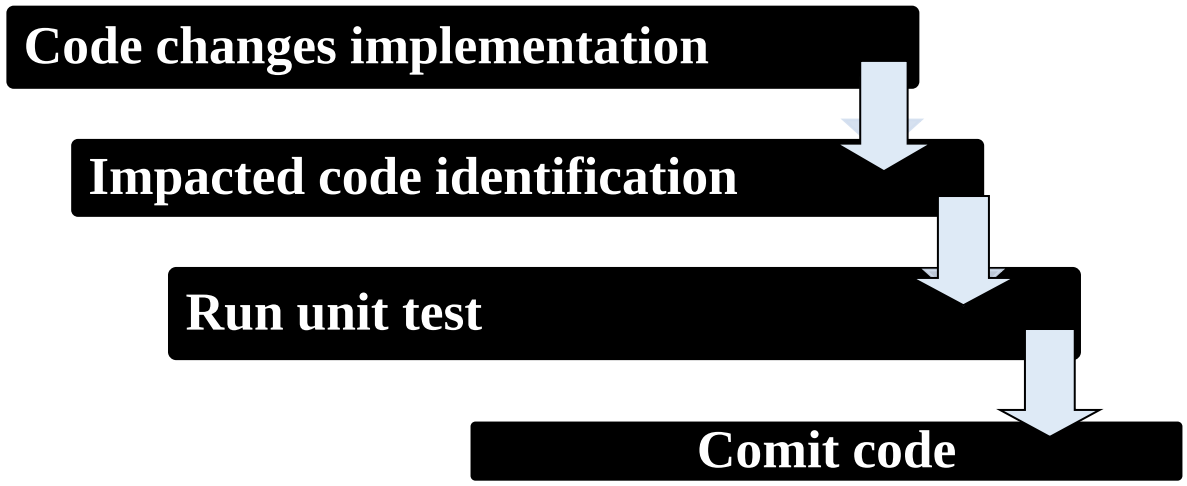


Figure 2- healthy practices followed by the developers before code commit

In order to alleviate the burden on developers, particularly if they might not be thoroughly experienced in all aspects about a code, different tools may be utilized to perform smart impact analysis with respect to the changes made, in identifying any other appropriate changes by examining commit history, and also to identify any regression tests relevant to the changes.

While performing change impact analysis, both preventive maintenance and code evolution becomes crucial. Also, it is always preferred if the code evolution is done by the same developer who developed the code, as he will all the relationships that exist in the code and it helps in reducing the likelihood of errors . However, it is never the same developer in most of the cases; hence we take help of the various tools to make the developer familiar with the relationships and other details involved in the code.

2.2 Classification and Differentiation of Impact Analysis

In order to classify and compare Impact Analysis techniques, we need to have a framework to establish a relationship among the starting impact set, the actual impact set and the estimated impact set. Bohner et al developed four types of IA: Start Impact set and Estimated Impact Set, Granularity, Estimated Impact Set and System, and Estimated Impact Set and Actual Impact Set. The usefulness of this methodology has been evaluated utilising techniques for IA, like slicing of program and manual cross-reference.

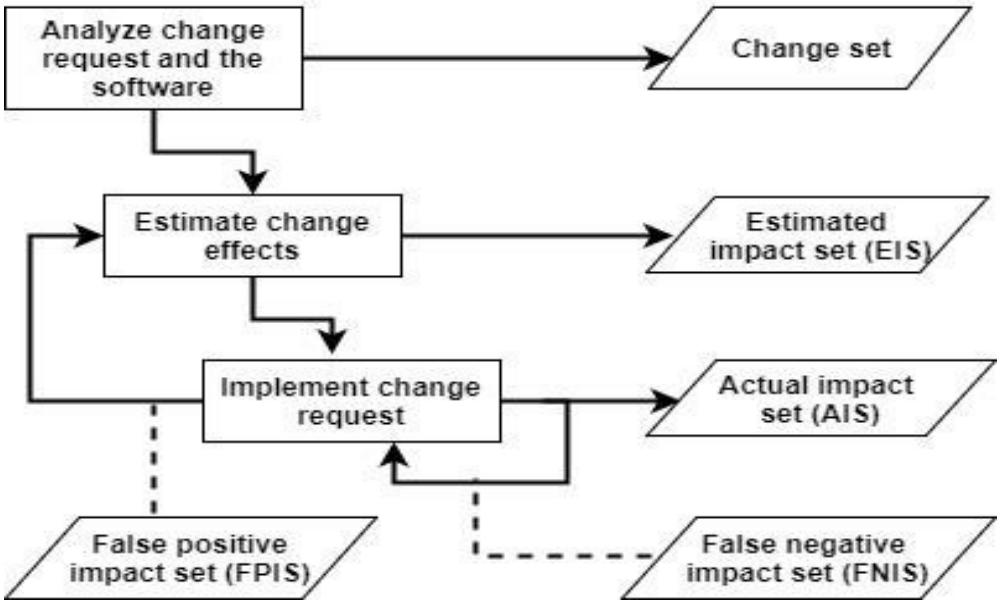


Figure 3- Change Impact Analysis Process

The IA framework can be classified as impact analysis applications, impact analysis parts and impact analysis effectiveness. Impact Analysis Application looks at how IA can be achieved using analysis approaches. This takes a look at the benefits the Impact Analysis approach provides. Impact Analysis Parts examines the type of the internal components and methodology used to perform the Analysis. While Impact Analysis Effectiveness investigates the properties of the resultant impact search, in particular how much they achieve the IA targets.

A taxonomy for classifying various approaches to the analysis of impacts was developed by Lehnert et al. Establishing on Bohner et al work[] which failed to keep all facets of Impact Analysis. The following classification methods have been established for the techniques:

- Scope i.e. various kinds of analysis done w.r.t. code.
- Techniques used i.e. the Impact Analysis technique that is used, the techniques normally used are program slicing, history mining and dependence graph etc.
- Tool Support i.e. the tools that help implementing the analysis approach.
- Practical Results i.e. the system size or accuracy.
- Level of details used for the approach i.e. detail level of the by-products of the software development, modifications and results.
- Programming languages supported, (like C, Java) by the approach.

Precision and Recall holds special important in the subset criteria. Precision is the intersection of Estimated Impact Set and Actual Impact Set to the Estimated Impact Set. While Recall is the intersection of the Estimated Impact Set and Actual Impact Set to the Actual Impact Set.

The established taxonomy by J.W. Wilkerson [] is centered on the classification of the algorithms, this is done by evaluating the way algorithms work at a initial level. Top level categorization began by finding the techniques that may have a significant impact on the code. Direct impact can be classified as: additional impact, modification impact, and deletion impact. While the indirect impact types are lookup impact, invoked method impact etc.

Techniques of Impact Analysis-

The program slicing is a method to improve codes by concentrating on selected aspects of semance. The slicing method erases all parts of the program that are calculated to have no impact on the semantics. It is used for debugging, re-designing, system testing etc.

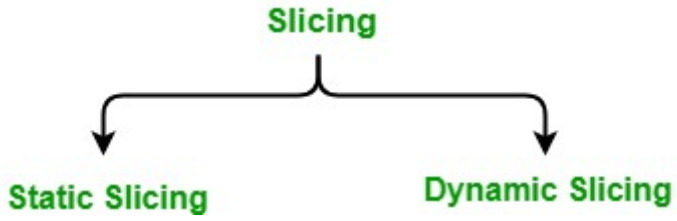


Figure 4 – Program slicing types

Types of Program Slicing –

- Static slicing-
    1. A static slice includes all statements that can influence a variable's value for an execution of a program at any stage.
    2. In general, the static slices are larger.
    3. This takes into account the every execution of the program that is possible.
- Dynamic slicing-
    1. Dynamic program slicing includes all the statements which can affect a variable's value at any given point for a specific program execution.
    2. In general the dynamic slices are smaller.
    3. Generally, it considers just a specific implementation of a program.

Slicing gives an insight into measures of stability and coupling, where coupling is the degree of dependence of software modules on one another. Also, this is usually determined by the given module's input and output information. Okulawon et al.[] concluded that program slicing improves the precision of program code. Wust et al.[] noted that the coupling measurement can be used as a way to identify the ripple effect in an OOS. Program slicing also provides cohesion metrics i.e. the degree to which program code elements relate together [][]. Meyer et al. [] carried out an empirical analysis employing program slicing for the coupling and cohesion measures.

Moreover, there are other practical applications of program slicing also, such as in re-engineering, program testing, refactoring etc. Frank[] conducted a survey research on different kind of slicing and classified the on the basis of their precision and efficiency.

The maintenance process involves change/modification management, IA, modification design, testing, and debugging and system integration. The sequential occurrence of these steps is explained in the figure-5 below,

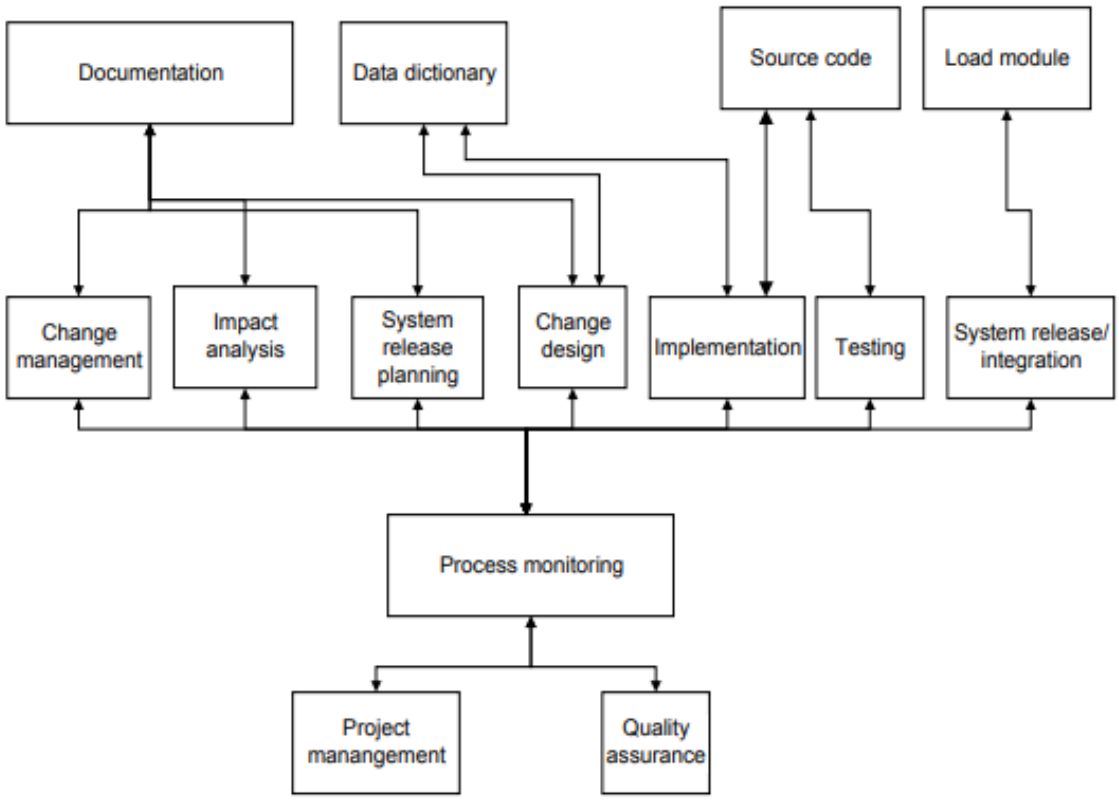


Figure 5- Steps involved in maintenance process

2.3 Impact Analysis tool attributes

IA information can be used to plan changes, implement modifications, accommodate specific sorts of changes in the software and trace the impact these changes. Thus making the potential results or affects of the modifications noticeable to the developer, making is simple to identify ripple effect and make precise and effective changes while keeping in the cost of the change. It identifies the parts of the code that requires a repetition of tests i.e. the regression test during the software maintenance phases. Overall, it saves time, efforts and money.

Various attributes or features of the tool support can be classified based on these measures or properties, as stated below,

- **Program Slice** or Impact Set visualization provides developers with the capacity to recognize a slice of the program at a particular instance in the program. Numerous tools which provide this ability have a method of showing the user, the program slice.

- **Dependency Visualisation** produces a graph for the software developers to better understand the dependence between the artifices and the codes.

- **History analysis** using repository mining,

- **Regression** is used for test selection by finding all the affected regression tests, given an input of changes.

By taking a look at all these features and prioritising them based on the task, the user or the developer can decide upon the tool that he is wants to you. At times it is noted that regression test has be proven to be more effective than the given impact analysis approach and algorithm.

CHAPTER-3

Survey of IA Tools

3.1 Methodology

In the following section, we will focus on the methods opted to perform the survey on the various impact analysis tools, for this purpose we will be performing a structured literature survey based on Barbara's work [].

The main aim of this literature review will help is understand various features of each tool and based on that we will be further able to classify them in to different categories.

3.2 Structured Literature Analysis

In order to conduct a structured literature analysis, we formed a few research questions –

RQ-1: What the impact analysis tools developed so far?

RQ-2: What functionality and attributes do IA tools offer?

RQ-3: How many opportunities subsist to develop tools for impact analysis which helps in maintenance phase of the developed software?

The above Research questions are answered based to a structured literature analysis, that is performed based on the following steps –

- Literature Hunt –
  The literature hunt is based on searching for the relevant information in form of publications, articles and blogs, this is done by giving appropriate inputs into the data database. We also used other sources like YouTube to get an even more clear picture of the impact analysis use, its types, tools and the attributes of various IA tools.

- Selectivity-
  The publications and reports are made even more selective in order to have a focused study by the criterion of convergence and divergence.

- Retrieval and interpretation of data-
  The information extracted from the publications and articles are inspected and classified into different formats based on the survey questions. The data analysis involves inspection, transformation and modelling of the data retrieved.

Various database repository used for the **literature hunt** are ACM, IEEE Xplore digital Library, Web of Science, Elsvier, ScienceDirect , Scopus, ReseachGate and Springer.

Some selection criteria were established to refine the set of selected articles even more , to a selective study. Initially, we regarded publications which concentrated on describing and portraying tools. Sometimes in cases, tool was created to highlight a specific of IA approach or a slicing method; that was permitted.

Alternatively, separating tools and methodologies from one another is critical. The distinction that paper identifies is due to is context. A methodology is a means or process of performing a specific function in the digital environment and a tool is an application that executes methods. A methodology is autonomous of the language or interface used for coding.

The selection criteria used includes

1. A tool that uses IA technique.
2. Tool that can be used before the final modifications are make in the actual framework that can used at the program-level.
3. Tool that aids the user in identifying the consequences of a change before it is actually made.
4. It should be presented in the publications that are written in international language i.e. English.
5. It should be able to work on the programming languages like C.

All the identified publications & tools were recorded for data retrieval and interpretation process, and each tool were differentiated from the rest based on its attributes. Such attributes apply to the tool's functionality and abilities, as well as to the tool within. The figure-6 below represents a typical Impact Analysis process,

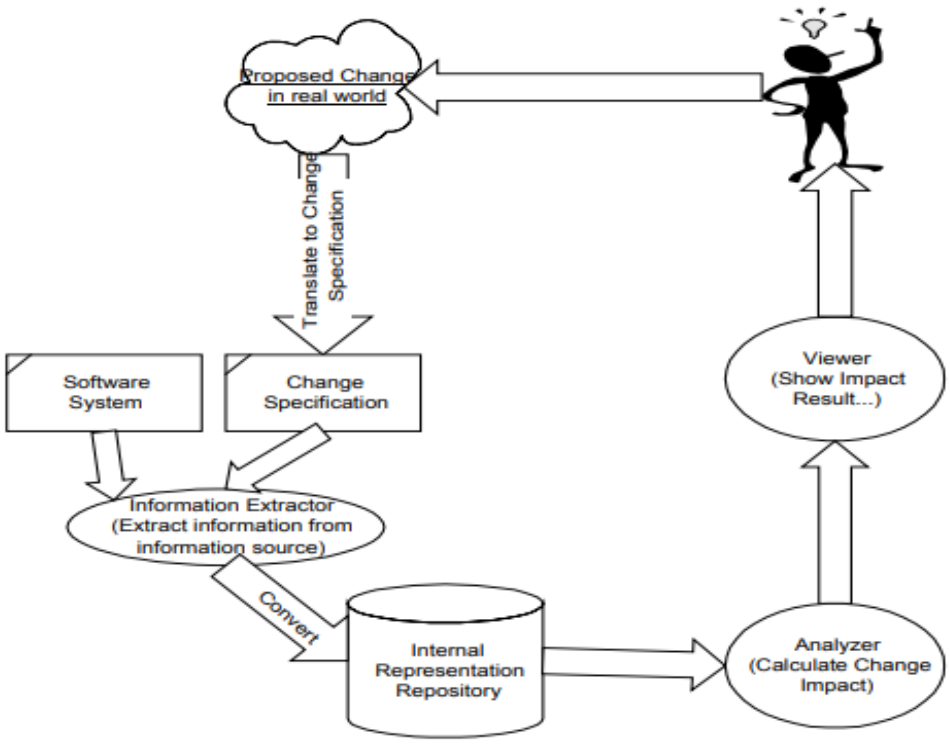


Figure-6 IA process

This process involves-

1. Transforming a suggested modification into a system-wide change.

2. Conversion of the retrieved information from a source into a repository representation.
3. Analyse the impact or consequence of the change or modification on the system.
4. Repeat the steps 1-3 for all the other proposed changes, in order to have a fair comparison among them all.
5. Create an estimation of the resources required in order t accommodate the change.
6. Interpret the implementation cost, advantages and the drawbacks of the change, just the way it is done for a whole new application.

3.3 Developed Framework for categorisation of IA tools

The data retrieval can be made easy by introducing a framework in order to clearly evaluate tools. This should be done because currently there is no such framework available. Many systems exist for characterization and comparison of IA techniques[4][12][19][22]. This classification system was influenced by these taxonomies and structures. Nevertheless, it is worth remembering that the methods and tools are different in nature. For this context, we'll concentrate on strengths of the tool alone, instead of concentrating on the methodology used by the tool.

Above mentioned model is composed of the following: generalized knowledge about tools, usability of tools, types and methodology of the tool used in impact analysis, and functional capabilities of the tool. Each of these is described in the following sub-sections.

3.3.1 Generalized knowledge about the tools

Basic information related to each tool was gathered. It is done in order to get a basic understanding of when it was being built as well as to know its usability. Also reported were the publishers of each tool, and the server that published the original paper.

We retrieved the following data about each tool, to get a better understanding of each one of them-

- When was the tool created?
- When the tool was last reconditioned?
- What all programming languages can be used by that tool?

It should be notes that if we could not track when the tool was reconditioned last, it was marked as not applicable and we did not end up assuming that its development is no longer taking place, without any specific evidence to support is assumption. Moreover, it should be kept in mind that may be some human errors involved while extracting these information, especially is the tool under study is not open to use by everyone and has accessibility restrictions.

Also, the tools that are commercially available in the market, the year when it was last reconditioned is considered to the year in which this survey is conducted that is 2020 as such are reconditioned more frequently.

3.3.2 Usability of a tool

It is very important to determine the usability of a tool because only if the tool can be made available for use to the users, it is considered to be usable. Once a tool has been developed it should be accessible to the users for both commercial and general purposes to make the best use of it.

To estimate the usability of a tool, following features should be looked at,

- The tool must be available for outside use without any restriction.
- The tool should be made available in the commercial market too.
- The tool has to be publically available.

Whether the tool is publically available or not may be doubtful at times, and it can be clarified by reaching out to the contributors of the development of that tool if necessary.

3.3.3 Types of IA

Each type of impact analysis tool using a different technique and we need to explain and categorize each tool based on the methodology or technique used by them. This is necessary as the developer should be aware of the underlying method on which a tool is based, so as to get the best out of it.

The tools can either be based on static or dynamic IA and they are be further classified on the type of method used by them such as

- Slicing of the program
- Data extraction
- Information gathering
- Detectability
- Statistical approaches
- Tracking implementation
- Code dependence graphs

3.3.4 Functional capabilities of the tool

It was observed that many of these tools have some attributes in common, such as

- They could perform the repetition test, also known as the regression test.
- Program slice can be presented by each of them.
- They were able to provide the EIS, AIS, SIS.

3.4 Research

Based on the research methodology explained above, we did a study in which we thoroughly visited fifteen research article and we came up with the following findings about each tool, as explained in the sub-sections below.

3.4.1 CodeSurfer

- CodeSurfer make the code readable and more understandable for the user.
- It gives you a complex image of your code and its effects.
- Its features are –
  - Pointer Analysis estimates the relation between the variable, even when they are affected by the pointer.
  - Call Graphs identifies the way different functions related to each other.
  - Global Variable Analysis defines how a global variable is utilized by a function.
  - Impact Analysis is used to analyse the potential consequences of a change, in the overall system.
  - Whole Program Analysis calculates the code relatedness among different files.
  - Dataflow Analysis check for the flow of data, i.e. the origin of the value of a variable.
  - It has one add-on feature also that allows it to analyse all the properties along all paths of a program code.
  - It also provides an optional programmable API.
  - CodeSurfer is made accessible to the educational institutes, free of cost.
- History Review
  - It was developed for the research purpose at the University of Wisconsin.
  - It was developed from 1996-2000.
  - It had the basic features, flow-control, forward slicing, backward slicing and the relevant parts are highlighted while slicing to provide a better overview to the user etc.
  - CodeSurfer 1.0 came in June 1999.
- Drawbacks-
  - Earlier it had no inline assembler hence the code was not clean, rather irregular.
  - There is an upper limit to the number of lines of the code, it is approximately 100k

    .

3.4.2 eROSE

- Principle –
  - Whenever a new modification or change is committed in a large software system, there is a need to make some other changes or updates in the system to avoid possible problems or bugs in future.
  - eROSE works similar to the shopping applications like Flipkart, Myntra etc. The shopping applications suggest their users, various products to buy base on their previous purchases. Similarly eROSE plug-in for Eclipse provides suggestions to the users based on the system history.

- Features -
  - eROSE is capable of finding out item coupling which is not detected in program analysis.
  - It suggests modifications, changes and updates in order to avert any future errors or bugs, based on the information stored in the version history.
  - Improved code navigation is achieved by using eROSE because for any small modification in the source code, it proposes or changes deduced from the version archives.

- It also helps in preventing bugs and errors.

3.4.3 Executive-After Tool (EAT)

- EAT was developed to highlight the benefits of dynamic IA over Static IA.
- EAT is three - stage process, edited in java programming language: an implementation framework, runtime controllers, and an analyser model.
- The details of the method comes from technique level, however the modifications can be include the changes at the source code level.
- EAT was created to analyse collectEA technique, which is both time and space competent.

3.4.4 FaultTracer

- FaultTracer[22] is used as a fault analyser tool. It performs CIA and regression test fault analysis.
- It is openly available and its front-end is used as a plug-in for Eclipse while the back-end is executed as Ant tasks, that is used to make the call graph process and identifying and raking of affected tests and affected changes automatic.
- FaultTracer performs the fault analysis by taking previous and current versions of a source code along with a regression test package and then locates the altered or influenced tests, taking into account the atomic changes that are applicable to the affected changes.

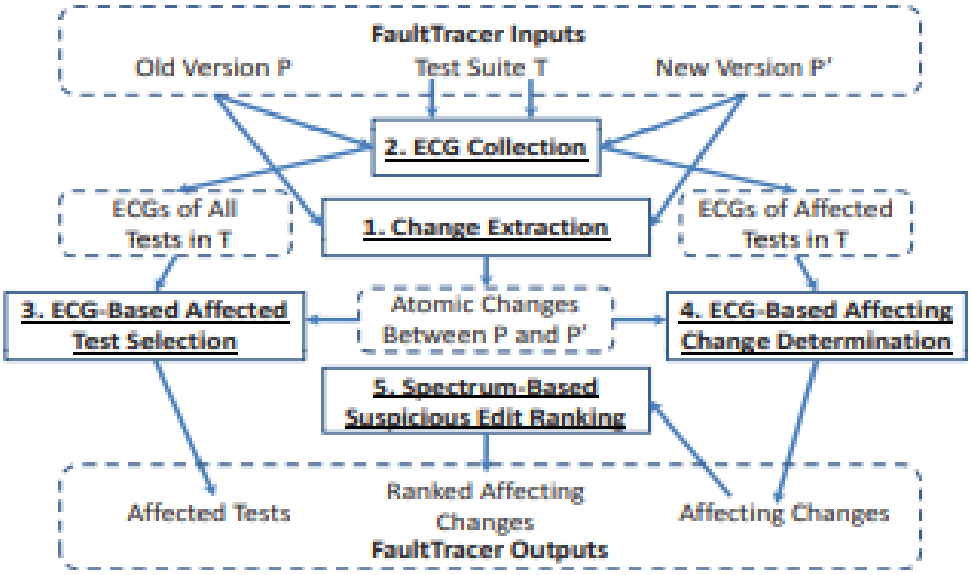


Figure 7- FaultTracer Basic Architecture

- As can be interpreted from figure-7, the FaultTracer does fault analysis by taking into account old and new versions and a test suit to give affected tests and affecting changes at the output.
- Any changes in the program code during the evolution of the software have tp be verified. This validation is done by using regression test suit.

3.4.5 Imp

- Imp is available for the change impact analysis of the source codes edited in C or C++.
- It is implemented as plugin for Visual Studio.
- Its focus is on improving the execution and precision problems related to static program slicing.
- It can be used to do risk assessment, dependence and consequence analysis as well as to perform regression.
- It is a user friendly analysis tool that provides the highlights of the analysis to the user or developer.
- It can be viewed as an improved extension to the CodeSurfer CIA tool, that gives better precision and execution performance.

3.4.6 ImpactViz

- ImpactViz is aimed at visualising the class dependencies and analysing the change impact in a code.
- As it is known, that OOS is based on reusing a code multiple times, thus an error generated in one part of the code can be identified in other sections of the code, such class dependencies can be observed by using ImpactViz.
- The depiction of ImpactViz is shown in figure-8, here the classes are presented by nodes and the method call among the classes is defined by the edges. One class can call a method in another class by pointing towards it.
- Thus it uses a graph design to highlight the calling method.
- It also has a feature that allows it to give different colours to each class, so as to easily differentiate between them.

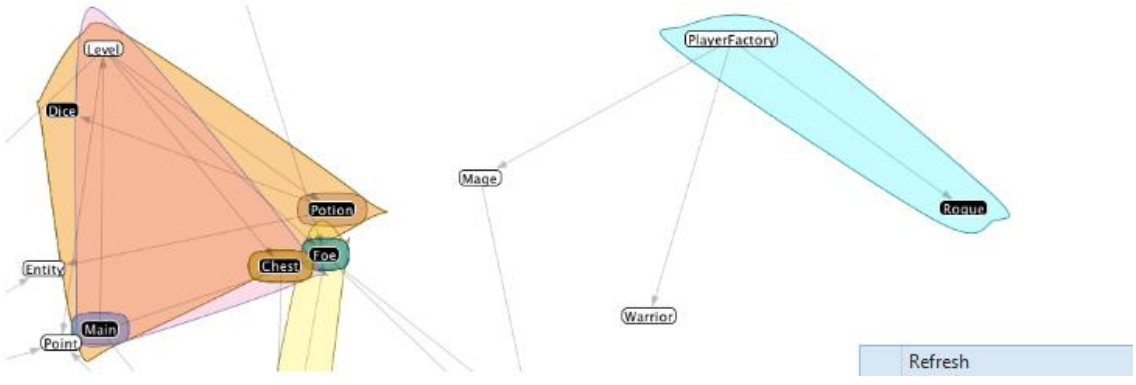


Figure 8- An example of dependency visualisation done by ImpactViz

3.4.7 Frama-C

- It is open source software used for change impact analysis.
- Frama-C stands for framework for modular analysis of C programs.
- Frama-C performs the code analysis of C software.

- It can be used on Linux, UNIX, Mac OS X and Windows.
- It is based on static program analysis i.e. the source code is analysed with executing it.
- It helps you understand a source code written in C. It helps the user to identify the set of values, does program slicing and navigates you through the code.
- Frama-C allows the user to identify and analyse the impact of forward slicing.
- It is only analysis tool that is up to date on various community platforms such as Wikipedia, github and has a lot of publications and work.
- Apart from impact analysis plugin, other plugin that are available are data-flow browser plugin etc.

3.4.8 ImpactMiner

- ImpactMiner [] is an impact analysis tool that uses database extraction of source code. It provides these IA techniques: repository data analysis, Dynamic implementation tracing, and data retrieval.
- It can be used for the analysis of Java source codes, by executing it as Eclipse plugin.
- It can be executed by using any one of the IA techniques mentioned above or a combination of them. Hence it can analyse the impact set using an adaptive combinations of these three different techniques to get the best results.
- Out of the three techniques, the data retrieval method needs a lots of time and efforts and still the results are not precise, where as repository data mining or history extraction and dynamic implementation analysis only require time and efforts while collecting the data or information and gives very precise results.

3.4.9 Indus

- Indus [] is an open -source software, it provide the Java program slicing and offers analytical techniques to assist programmers in improving Java programs.
- It has following key functions: forming a set of functionality for static analysis, and the Java program slicing.
- It also provides forward slicing, backward slicing and complete slicing to the user. The user has the option to select any one of these.
- It can be used as an API.

3.4.10 JArchitect

- It is used for the static program analysis (i.e. performed without actually executing the code) of the Java code[23].
- It provides the visual dependencies between the design structure matrix and dependency graphs.
- It has the capability to perform restructuring at scale and find repetitions in the source code structure.
- Apart from the dependency graphs, JArchitect can also provide use of other graphs, like coupling graph and cycle graph etc.
- Using the dependency graph, the user gets a better idea on how the relationship or dependencies in the source code impacts other methods.
- It also provides a fourteen day trial period for the users before purchasing its commercial license which is available in 2 different types.

3.4.11 JSlice

- JSlice is implemented in Eclipse plugin.
- It is an open-source java dynamic program slicer used for program debugging and understanding.
- It works on adaptive program slicing approach where is allows slicing only the final executed statement or all the statements of the source code.
- It compresses the traced program execution by slicing the program and then analyses the compressed form.
- Its latest version JSlice 2.0 was made in 2008 and has been updated later.
- It uses kaffe virtual machine in order to trace the execution and then compress it for the analysis.

3.4.12 Jripples

- The evolution of the software requires adding new attributes and functions in to the software[20]. In order support improvement and maintenance in a software, Jripples provides the developer a support that makes further changes and improvements in the software easy and structured.
- It is an intelligent tool that navigates through the code and records the irregularities in the program and then reminds the developer to work on them. Thus the programmer can focus on the larger picture and the granularities involving algorithms are left for the too, to handle.
- It is implemented as Eclipse plugin.
- It takes care of the granularities like the propagation of the change or the modification through the object oriented software code.
- It first performs the data retrieval and then find the dependencies between different classes in the program, therefore it can be said that it works in the same way as the developer.

3.4.13 Kaveri

- It is implemented as eclipse plugin. It is a program slicing tool.
- It is an extension of Indus impact analysis tool and it provides improved slicing capabilities in comparison to the Indus as it hides way the unimportant details about the working and execution followed by the tool.
- It can be utilised by the users to navigate through the code to identify any changes.

3.4.14 SLICE

- SLICE[32] is a program slicing tool that uses dynamic techniques to slice the C program.
- Also, the dynamic slicing can considerably reduce the size of the slices in comparison to the original program.
- It can support both forward slicing and backward slicing.

3.4.15 Ripple Effect and Stability tool

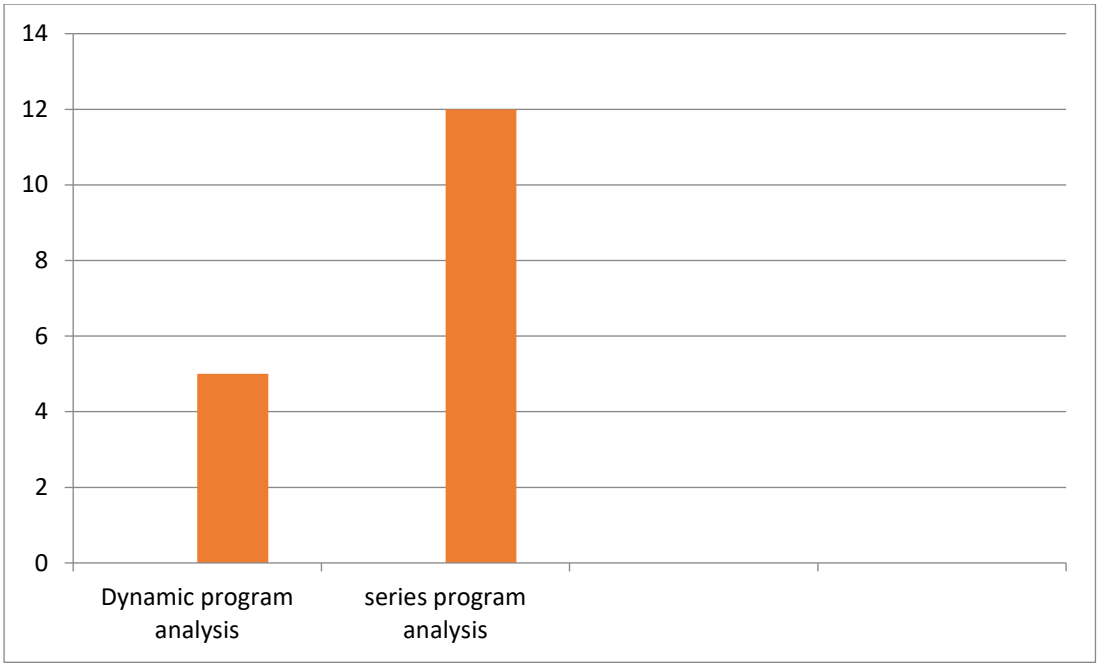
- As the name suggest, REST estimates the ripple effect that a change or modification will have over the entire code or program[34].
- It can be used for the code written in C language and it is used to locate a change and then estimate its impact.
- REST is used to evaluate the stability of the source code written in C.

3.5 Analysis tools- Availability and types

Table 1- Tools availability and methods used

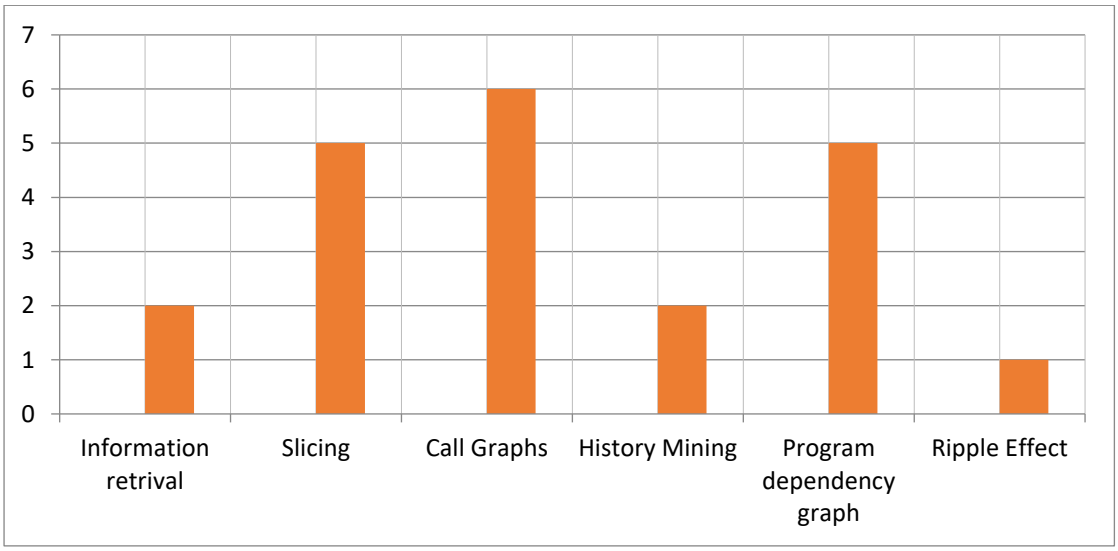
Tool	Dynamic program analysis	Static program analysis	Open Source	Commercially available
CodeSurfer	No	Yes	No	Yes
eROSE	No	Yes	No	No
EAT	Yes	No	No	No
FaultTracer	Yes	Yes	Yes	No
Imp	No	Yes	No	No
ImpactViz	No	Yes	No	No
Frama-C	No	Yes	Yes	Yes
ImpactMiner	Yes	Yes	Yes	No
Indus	No	Yes	Yes	No
JArchitect	No	Yes	No	Yes
JSlice	Yes	No	Yes	No
JRipples	No	Yes	Yes	No
Kaveri	No	Yes	Yes	No
SLICE	Yes	No	No	No
Ripple Effect Stability Tool	No	Yes	No	No

The analysis shows that out of 15 tools studied, static program analysis is used more frequently, 12 tools are reported to be using static analysis, while only 5 tools use dynamic analysis. The data can be seen below in form of a graph.



Graph 1 - Program analysis technique analysis

Moreover, each tool provides a different set of attributes and functionalities and they are be categorized based on these methods. Following has been displayed using a graph, as shown below. Here is Y axis represent the number of tools(count) and the X axis represent the specific attribute or functionality.



Graph 2 - Features available in IA Tools

CHAPTER-4

Change Impact Analysis

All the security standards suggest change impact analysis (CIA) in the development process of sensitive applications and software. Even though the CIA can be considered as a very basic activity, practical studies about its execution during the development of a sensitive application are still missing. In this study, our objective is the presentation of an automated system in the context of the CIA for sensitive applications based on professionals' overviews. Our analysis proposes that computer and software engineers spend 60 – 80 hrs on the CIA per year. However, this time duration differs significantly from the phases of development. Moreover, the respondents described different implications to the CIA and supposed the significance of the CIA differently. We present important CIA challenges and several research ideas for the CIA. Additionally, this research work only presents primary analysis, our work also draws attention to practical importance to the CIA.

Sensitive software and applications change during their life cycle with more effective security techniques and standards. These changes also have impact on the systems in which these applications and software run; therefore change impact analysis is required, which highlights the results of any change in the system. The CIA also predicts the modification in the system for compliance with the change [1]. The primary objective of the CIA is the security assurance, and also suggested by the safety standards like IEC 61508 describes that "at any stage of a software lifecycle, if any change is required then CIA will detect the software modules that are going to be affected by this change, and earlier security practices for lifecycle will be revised." More similar standards that the designer needs to stick to while developing a sensitive application are ISO 26262 that is used for automotive industry, IEC 62304 that is used in medical industry etc.

Sensitive software and applications are those systems which are life-critical, i.e. they aim at creating safe systems. Any failure in such systems can be detrimental and can result in loss of life and property. A few examples of such software systems are spaceflights, railway signalling systems, airbag systems, braking systems, air traffic control systems etc. Some other traditional examples of safety-involved software or sensitive applications are compilers, requirements traceability tools, simulators, and test rings. The software that is used for generating the data used by sensitive applications is also considered to be a sensitive application. Moreover, some technologies are required to go beyond avoidance of collapse and they are often used in medical industry such as dialysis machines, heart-lung machines, medical imaging devices etc.

As the system complexity is increasing at a very fast pace, safety mitigation measures becomes an essential requirement for safety-involved sensitive applications, such critical systems should be designed such that they fail safely, i.e. the system should not turn off immediately when the power goes off, for instance, in case of a heart bypass machine. Moreover such sensitive softwares are constantly changed during their life time; hence the analysis of the result of such changes on the overall system performance becomes necessary. In order to identify the modifications required to be made, to accomplish the change, the use Change impact analysis becomes essential.

Sometimes, the CIA is considered as a difficult task to perform due to the sensitive systems [1] [2]. Insufficient CIA activities have been the reasons for vulnerability exploitation in the

past [3]. Now, there is a time to get benefits from new CIA techniques, knowledge, and practices to more economical security assurance, to avoid risks and mitigation of attacks.

Despite the importance of the CIA for security-critical applications and system, there is a limited research study for the CIA techniques and practices. The available research studies mostly focused on non-sensitive applications and systems and analyzed data from the previous work. For instance, Rovegard et al. [4] took views from software engineers to analyze the significance of the CIA. Though Borg et al. analyzed the reports based on an industrial control system [5]. Some implications have been presented in [6] regarding verification and validation of requirements and on traceability in [7]. Nair et al, [8] presented with the safety evidence management methods, highlighting some points about change management.

We performed a survey on sensitive applications and systems for the CIA and asked questions from software engineers to get their views on security practices to develop a sensitive application. We interviewed 14 software engineers in two teams of assessment from two different organizations working on sensitive software projects. Our objective is to help technical decisions while developing cyber-security applications or physical systems, in which the CIA has its importance. To conduct these interviews for the sake of research, we mainly asked the following three questions-

(Question-1) How difficult is the CIA work activity?

(Question-2) What is the software engineer's behaviour towards the CIA?

(Question-3) How to support CIA for the completion of engineer's tasks?

4.1 Research Methodology

We formed a questionnaire that can be used to survey multiple industrial sites, and it should not deviate from its context. The questionnaire is constructed such that the three questions mentioned in Section I of the paper, are addressed in a detailed manner. We took two development teams establish assessment units, referred to as Team 1 and Team 2. Figure 1 describes the research process, where smiley presents the number of researchers involved in each process.

The following steps where involved in the research methodology; (1) Three researchers repeatedly designed the questionnaire in a case-study protocol format. And all the phases in the design were keenly reviewed by experienced researchers. We used an interview instruction set (available online [9]) that assists to ask open and close-ended questions. We started with open questions and ended interviews with the time glass interview model [10].

(2) The process of interviews from engineers is processed in the English language. For the reason of secrecy single researcher from the team did interviews. (3) The same researcher did the assessment on conducted interviews and forwarded them back to the interviewees for acceptance. (4) We interviewed fourteen professionals in team-1 of whom nine are software developers that develop sensitive applications and write their documentation. We also interviewed an R&D manager specifically, two security engineers, and two senior developers. On the other hand, in team-2, we interviewed eight professionals, four software developers, two senior developers, one technical manager, and one product manager.

(5) For the primary analysis step, the acceptance report is processed further and long answers are divided into the bullets. Unnecessary answers were removed and the analyzed report was filtered further for the coding and debugging context. The acceptance report was based on the following scheme explained as follows; A 2-D coding scheme that used the intent of CIA and its importance along the two axis, taking both negative and positive range of values to show the result was followed in Pre2.a; while Pre2.b used the frequency of conducting CIA activity as a measure; a time axis was used in Pre2.c; Pre2.d coding method deals with some of the more difficult challenges of CIA; the time and the difficulty level after modifications addressed in Pre3; and Pre4 open coding is based on the support system provided the change impact analysis detection. Each of these the coding scheme explained above are represented by their respective IDs (used for the convenience of understanding) as listed in the Table-1. Here "Pre" notation is used to emphasise that it is the pre-reporting process.

(6) Finally, in reporting process, we finalized the report, for the matter of secrecy and confidentiality; we hide traceability from the answers.

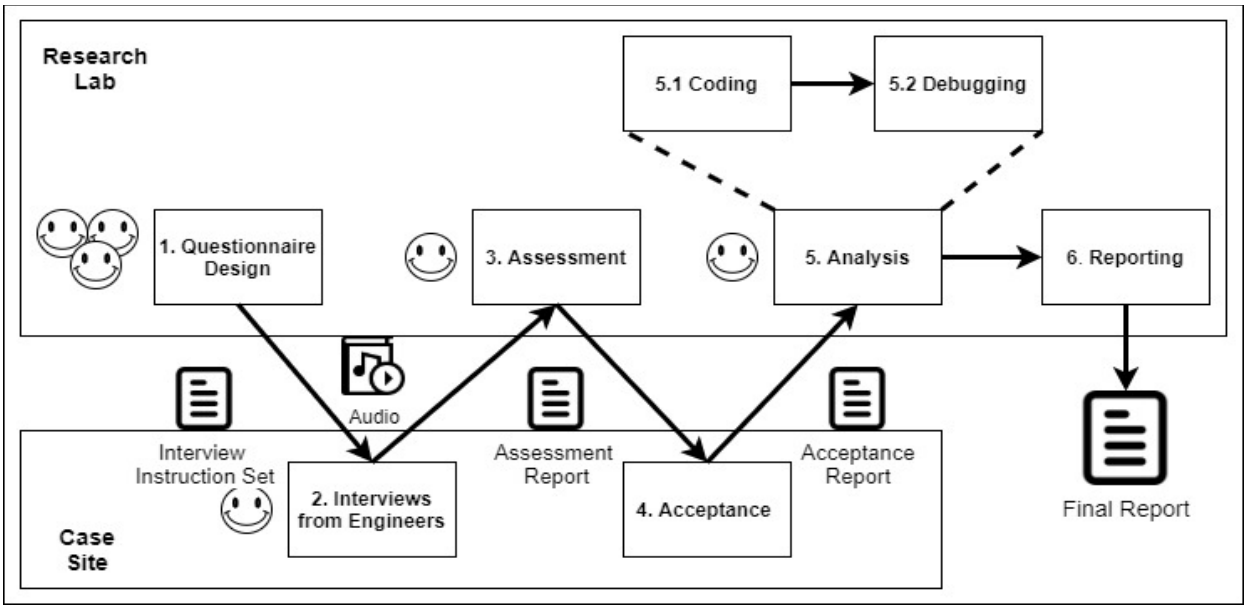


Figure 9 -Layout of research process

Table 2- Categorization of questions to specific parts of the interview instruction set

Questions	ID	Interview Instruction Set
How difficult is the CIA work activity?	Pre2.b	Do you do CIA activity daily/weekly/monthly?
	Pre2.c	What is the duration of time spent on the CIA?
	Pre2.d	What are the CIA problems?
What is the software engineers' behaviour towards the CIA?	Pre2.a	What are your views on CIA activity?

How much the CIA is supported in the completion of engineers' tasks?	Pre4	What type of assistance do you need in the CIA conduction?
	Pre3.a	Would you like to comment on the parameters, we have gathered from previous CIA conducted activities?
	Pre3.b	How much it would take to complete the CIA process?

4.2 Case Description-

Sensitive applications are developed to run on sensitive or highly secured systems. It is necessary to properly examine the source code before implementing it to the system, in order to develop good quality software. Additionally, detailed documentation should be formed for future reference and abstraction. A process of project completion needs collaboration from engineers because every project consists of over one million lines compiled in different computer languages.

After the completion of the development process, the organization requires to set a security case for external audit, to achieve a secure and safe system while operating in any given environment. The CIA analysis is a critical step for the security case. However, security engineers developed a CIA report template as presented in Table 2. The software developers follow the template, and conducted and documented their CIA before the changes to the source code are made. This template is derived from Klevin [11]**.**

Table 3- CIA template used in the company, derived from Klevin [11].

(Q-1) Is the problems security critical?
(Q-2) In which version of the software/application problem exists?
(Q-3) How sensitive systems will be affected by this change?
(Q-4) Enlist changed code modules and files
(Q-5) Which firmware is affected by the change?
(Q-6)After committing a change, which documents should be modified? (e.g,specs, architecture)
(Q-7) Which test to ensure security should be executed? (e.g., design/functional/sequence tests)
(Q-8) How long it will take to resolve the problem?
(Q-9) What is the major reason for the problem?
(Q-10) What all requirements and specifications of functions are required to be retested?

There are limited tools available in the organization to conduct the CIA. This conduction is completely linked with the issue management process. And all of the finished CIA reports are secured in an issue repository in form of attachments for issue reports. Moreover in scope to this research, there is a web interface available within the organization to access the issue repository.

4.3 Results

We investigate and present results the based on the survey, as will be discussed in the following subsections.

A. Research Question 1: How Difficult is the Change Impact Analysis activity work?

One of the key questions that require a well thought answer is the difficulty face by the experts while performing CIA. To understand the difficulties that professionals experience during the CIA work activity, we inquired about the occurrence of CIA activity by the professionals, the time required to complete the CIA activity, and the challenges experienced by the professionals during the CIA activity. Each of this will be discussed below.

The occurrence of Change Impact Analysis- All the professionals have stressful experience with the occurrence of the CIA activity that varies from daily to weekly to monthly, depending upon the development phases. Several professionals reported that they conducted the CIA activity daily, while some other professionals reported that they conduct it on weekly basis during the most extensive time of the development. A senior developer in team-1 estimated that an ordinary software developer may conduct 23 CIAs related to debugging every year. A security engineer in team-2 estimated a higher number: “in the high time of the project, a junior developer conducts CIA activity daily”.

The reason behind this variation is a phase-wise development process, in the initial phases of the project, the source code of underdeveloped software is churn, and the individual problems are not managed with change. Once a project completes some milestones and debugging and testing phases are initiated, the objective is to ensure the quality of the product, and all changes after this phase are related to debugging.

Effort in Change Impact Analysis - When asked about the average time spend on the CIA activity, five professionals responded with an average CIA effort. Figure 2 depicts a general view of the data collected. Similar to the occurrence results, CIA efforts varies significantly with the time required. A senior professional expressed his view that the effort may depend upon the complexity of the project phase, while others expressed that it depends on the documentation involved in the development process.

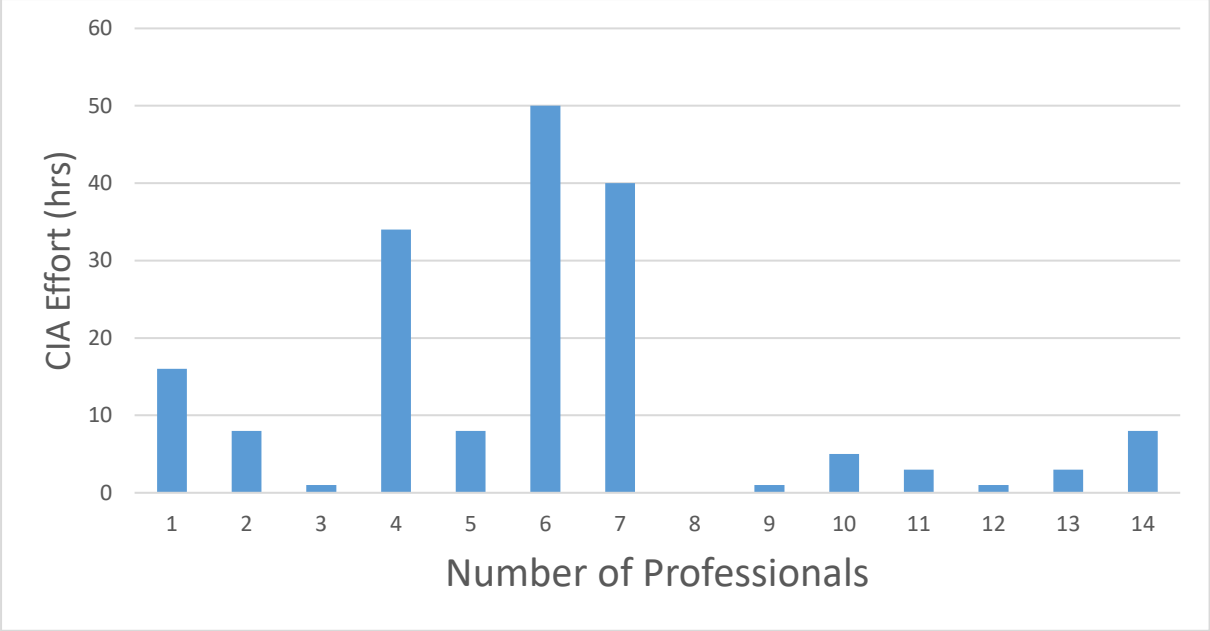


Figure 10- Professionals reported CIA effort.

Three professionals reported that the average CIA effort is an hour or two, four professionals expressed 5-6 effort hours, and three professionals expressed a day or two. On the other hand, three professionals reported that the complete CIA activity is the matter of not more than an hour, more specifically, 40-50 minutes.

Nine of the professionals reported that at least 35 minutes or less are required to finish a CIA, of which two reported 10 minutes or less. Two senior developers in team 2 expressed that swift CIA necessarily requires an hour or two. In the context of the maximum time, five professionals expressed 2-3 days, three reported 7 days, and a junior developer reported up to many months. We filtered three main reasons for this variation. First, it is difficult to delink the CIA from the general issue resolution; therefore, the professionals reported their answers differently. The explanation of this work involved in regenerating and understanding a problem connected with the CIA. To resolve a problem completely, professionals often need to set up a specific test environment, debugging, and testing source code.

Challenges to CIA- There are 30 major general and specific challenges reported by professionals presented in Table 1. It contains some questions that deal with management issues like Q-9 and Q-10. They mostly occurred challenge is related to motivation, like understanding the need of the CIA in the process of sensitive application development. A senior developer reported that “my primary task is to describe and motivate why we do CIA, however, we regularly remind ourselves why we do it. Furthermore, if the developer starts to know the importance of the CIA, it helps me in my work”.

The second most critical challenge is the information overloading; the senior developer reported that it is hard to understand the system due to its complexity. There is a huge number of documents presenting the system except for the source code. A software developer stated that: “looking for the right information is a major challenge and we need to find the key people and query for dependencies and other documentation, after all this, we do not get the answer.”

There are several other challenges worthwhile to mention, sometimes, software developers and engineers require previous CIAs documents that were processed long time ago. Resolving old issues is cumbersome and needs time and effort. Another important challenge is to establish trust in conducted CIAs to answer the questions in Table 1. Some professionals reported that they are not sure whether their answers are right or not, how could we evaluate it. Lastly, some interviewees said that the major challenge is CIA guidelines are not followed by the developers.

B. Research Question 2: What is the software engineer's behaviour towards Change Impact Analysis?

We highlighted the professional implications to "Change Impact Analysis", i.e., the behavioural connectivity towards the CIA. Moreover, we connect the implication to the impression of how significant the CIA is for the professional, depicted in Fig. 2.

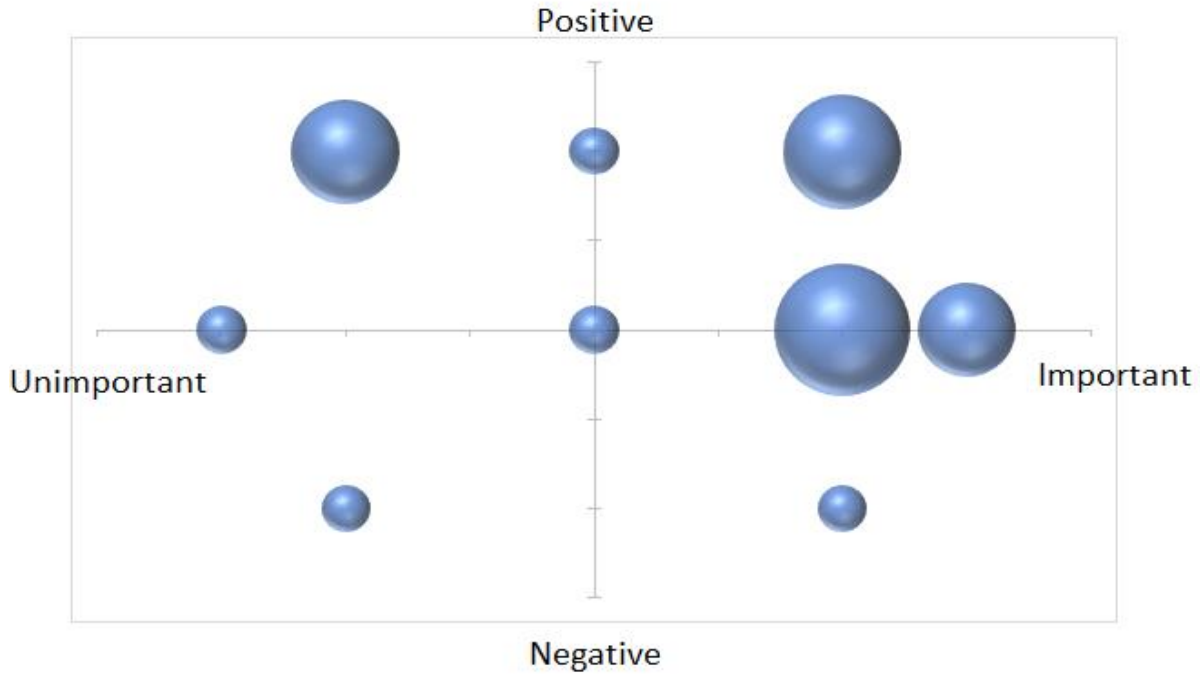


Figure 11- Professional's behaviour towards the CIA.

The professional's behaviour to the CIA described in all quadrants in Fig. 3. There are not many strong implications to change impact analysis. It was observed that a large number of engineers regard their CIAs essential. Several professionals reported positive associations, e.g., "a positive sign" and "shows that we do complex software engineering". Also, several professionals had neutral implications: "no values on a personal level" and "just part of the job". Two professionals reported negative implications. They consider the CIA is a too heavy or rigid activity, and need a better way to perform it.

Considering the professionals' supposed significance of the CIA activity to their work, all levels are covered. There are a range of responses from "for some issues, it is just worthless stuff, done for the process" to "professionally, it is very fundamental" and "side effects are

extremely important in our complex product". We observe no indications of any behaviour differences between the two groups. It is identified that, the senior professionals have a slight leaning towards the importance of CIA, when compared with the juniors. This comes from the fact that the seniors have seen more cases of side effects being caused as a result of source code changes, from earlier development work when the CIA was less formal, as explained by a senior engineer "we have seen many cases when fixes introduced bugs".

C. Research Question 3: How to support CIA for the completion of engineer's tasks?

CIA Support proposed by some professionals - When asked for potential methods to support the CIA, the tool support was most frequent answer by the software engineers. Three professionals proposed that the level of automation in the CIA can be increased by introduction of some tool support. A possible explanation is that professionals are already well-aware of the efforts being put on developing the traceability in the company and they believe it is not being utilised up to its potential. One proficient clarified how a tool may be traceable through a framework, from an input source code file, to a design depiction and then the resulting tests, at that point proceed up the abstraction layers to utilitarian details and up to the requirements. This permits to have the identification of the test straightforwardly from the requisites, as a tool-based arrangement that can near the regularly challenging loophole between the requisites and the test cases. Another scheme, provided by some lately appointed senior experts is that it's unusual to link a free-text string of a CIA report with some third party device, rather this can be done by using a tool developed inside the company, to have a way better command over the input.

Measuring CIA Support-The CIA history stored inside the issue tracking system was explored and the two possible measures were developed: TIME and MODS. These measures were estimated for the suitable interviewees and observations were presented during the interviews.

The first measure, TIME, is defined as "the time between a developer being assigned an issue and the first CIA report being submitted". Hence TIME focuses the effort needed to perform CIA, but none of the professionals considered TIME to be related to the amount of time taken to conduct a Change Impact Analysis. Six experts straightforwardly refuted this measure, while two of the professionals could not understand how to decipher it. The foremost doubtful views clarified: "it's definitely a question of priorities" "I work on several parallel products, and that measure can be anything", and "you have to measure when I begin making related changes". It can be concluded that TIME is a very disconcerted measure to be utilized in assessing and analysing the solutions that points at diminishing the time required to perform Change Impact Analysis.

Another measure, called MODS, can be expressed as the total number of modification on a CIA report after it has been first submitted. We recommended utilizing it as a proxy measure for the trouble faced while finishing a specific CIA; hence it focuses on CIA exactness. Three of the experts were somewhat positive to the measure, one of them explained that: "more modification implies a difficult CIA. And it implies that you couldn't promptly capture everything. While, opposite to this, two experts toppled the measure completely by communicating that various changes involve only the typos and copy-paste blunders.

Conclusively, TIME shows up to not at all be connected with the time required to perform CIA. While, MODS received some support, many also expressed that it is bounded by trifling changes to an extent that it can't be entrusted. While the two proposed measures can be considered simple to gather from the logging time stamps as well as the revision history of most of the systems, some enhancements are needed in both of them. While TIME must arise as a result of actual changes concerned with an issue and not based on when that issue is relegated, MODS should be modified to expel trivial changes like spelling and grammatical adjustments.

CHAPTER – 5

Proposed Method to select the approach of Impact Analysis

Lehnert[35] in his taxonomy provided various approaches that can be used to categorise the techniques of IA. This classification is made keeping in mind the following,

- The analysis scope that discusses what is to be analysed and this can be static, dynamic or online code and the architecture and requisites of the model.
- Granularities of each entity such as the variable, function, change ad granularities of result.
- Analysis style used i.e. global analysis, search based or exploratory,
- Technique or approach used like, program slicing, call graph, history mining, information retrieval, ripple effect, execution traces, program and message dependency graphs, probabilistic approach etc.
- Tool support
- Languages supported
- Scalability is essential to find how much the approach can be scaled.
- Experimental results such as the accuracy, recall, time and the size of the study set.

Lehnert successfully provided a categorization of the techniques to present different approaches available to a developer, however the developer still need to manually check, analyse and finalise the approach that best fits the software under consideration. This can be tedious and time consuming and the developer may end up losing the important time that he otherwise would have used in information retrieval, history mining and analysing other granularities of the program.

This motivated us to come up with a new proposed tool that can be used to automatically select the right approach based on the developers needs. It will provide the developer with the ability to select the desired and most fitted approach based on various parameters such as the analysis scope, language used, technique used, tool support and other granularities.

Thus, the proposed method can be thought as an extension to the Lehnert's taxonomy[35], as it attempts to make this approach selection method automated. The program for the proposed tool is written and executed in Python.

This program an run on any machine or computer, given that you have the following installed in your system.

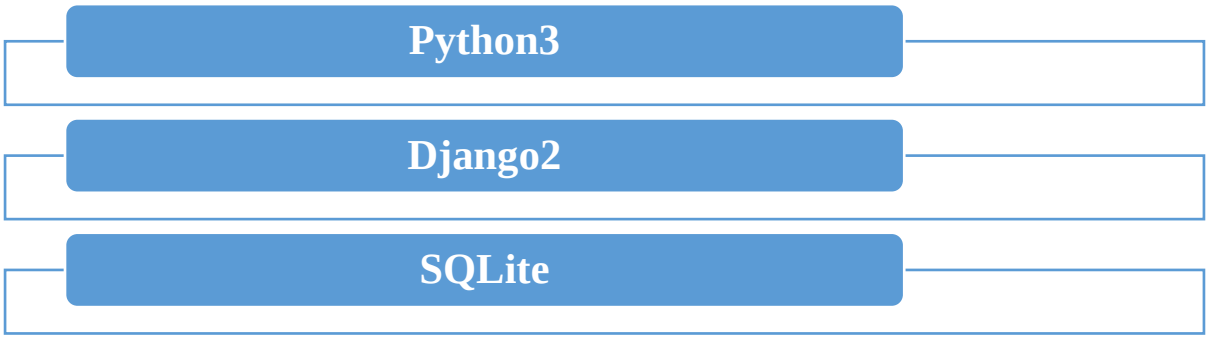
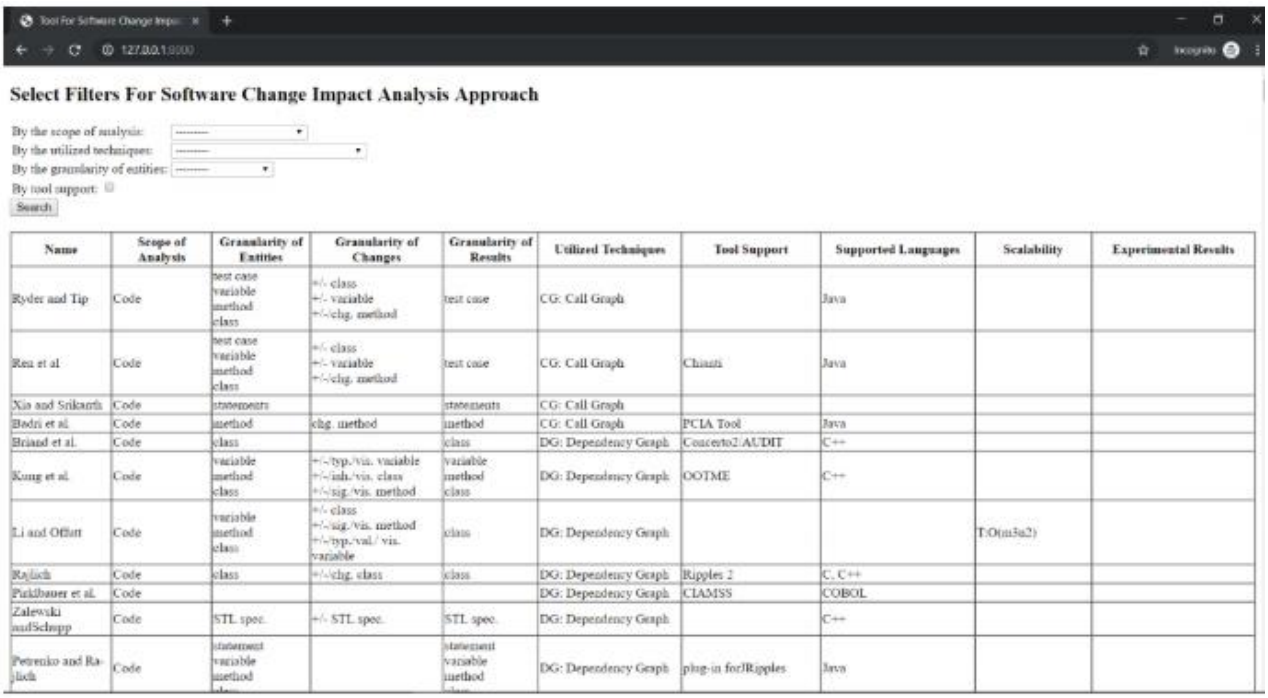


Figure 12- Installation required to run the program

Run the following commands for each of the mentioned tasks-
Initialisation of the Database- python manage.py migrate
Inserting json data to the Database- python manage.py initialize_data
To run the tool on the system python manage.py runserver

Table 4- Commands to run the tool in local machine



Select Filters For Software Change Impact Analysis Approach

By the scope of analysis: --------
By the utilized techniques: --------
By the granularity of entities: --------
By tool support:
Search

Name	Scope of Analysis	Granularity of Entities	Granularity of Changes	Granularity of Results	Utilized Techniques	Tool Support	Supported Languages	Scalability	Experimental Results
Ryder and Tip	Code	test case variable method class	+/- class +/- variable +/-/chg. method	test case	CG: Call Graph		Java		
Rea et al.	Code	test case variable method class	+/- class +/- variable +/-/chg. method	test case	CG: Call Graph	Chianti	Java		
Xia and Srikanth	Code	statements		statements	CG: Call Graph				
Badri et al.	Code	method	chg. method	method	CG: Call Graph	PCIA Tool	Java		
Briand et al.	Code	class		class	DG: Dependency Graph	Concerto2/AUDIT	C++		
Kung et al.	Code	variable method class	+/-/typ./vis. variable +/-/inh./vis. class +/-/sig./vis. method	variable method class	DG: Dependency Graph	OOTME	C++		
Li and Offutt	Code	variable method class	+/- class +/-/sig./vis. method +/-/typ./val./ vis. variable	class	DG: Dependency Graph			T:O(m3n2)	
Rajlich	Code	class	+/-/chg. class	class	DG: Dependency Graph	Ripples 2	C, C++		
Parikbauer et al.	Code				DG: Dependency Graph	CIAMSS	COBOL		
Zalewski andSchupp	Code	STL spec.	+/- STL spec.	STL spec.	DG: Dependency Graph		C++		
Petrenko and Rajlich	Code	statement variable method		statement variable method	DG: Dependency Graph	plug-in forIRipples	Java		

Figure 13- Screenshot of the proposed tool that provide the filter option also

This tool is designed keeping in mind the general data packet regulations that the companies have to abide to. The general data packet regulation is the regulation set that is presented by the European Union. This regulation set provides the privacy and data protection to the users. It is also concerned with addressing the personal information transfer outside the European union and European economic areas. GDPR is based on seven ideas for lawful processing of the personal data. These seven principles includes-

1. Justness
2. Usage limited to the scope of the purpose
3. Minimisation of the data

4. Precision
5. Limited storage
6. Uprightness and secrecy
7. Answerability.

Therefore, it is necessary to come up with the GDPR compliance tool.

CHAPTER-6

Conclusion

In chapter-3, we came up with a framework to classify various IA tools based on their accessibility, scope of analysis and the kind of technique used. In order to jump to this conclusion we did an extensive research, for this we read fifteen research papers and took help from different sources like Google, Wikipedia, Springer, ResearchGate, and ScienceDirect.

The results of this research provides a few important observations-

1. A stable and consistent source code is essential in order to have an error free evolution of software by carrying out appropriate maintenance because a slight modification in program can cause a lot of time, effort and monetary loss.
2. Use of IA tools will help analyse the impact of a change on the over-all system and hence it will save time and money, the two most important resources of any organisation.

The study in chapter-4 focuses on the professional's overview of Change Impact Analysis. We interviewed professionals in two teams of analysis in the context of sensitive software and applications run on security-critical systems. We concluded that efforts and time require to complete CIA activity vary, that dependent on the phase of the underdeveloped software with the complexity of required change. According to our results, software developers and engineers working in a sensitive organization spend around 60 – 80 hrs on CIA activity. Some senior engineers also shared their experiences that the CIA can take 10% of the overall project's time to solve a normal problem. We also mentioned some CIA challenges that include communication with the team working on a project and documentation of developed source code. We also concluded that the CIA is a significant but expensive activity in the sensitive software development environment.

We also observed the professional's behaviour towards the importance of the CIA. We also proposed CIA improvements that include the usage of additional tools, optimized traceability, individuals, and training programs. It is a little difficult to measure the value of CIA quantitatively because it is a secluded activity; however, it is well connected with development and management problems in a security-critical project. There is a need to analyzed CIA activity for quantitative measures in the future.

Our primary analysis also has some limitations; the professionals are obliged to follow the Non-Disclosure Agreement (NDA) due to the security reasons and policies of the organization. The interview and assessment activities were done by the first author (see Fig. 9) the acceptance step is done by another researcher. Additionally, we are focusing on the acceptance of our analyzed report. Another limitation to the experiment is that the results are concluded on the views of a limited number of practitioners; therefore, we plan to add more number of participants to the study to further increase the validation of our results. Moreover, the survey experiment is conducted only in one language i.e. English, this may act as a language barrier to some of the potential interviewees, thus we further plan on taking the interviews in more than one language, involving people from different countries, to overcome

this limitation.

While evolving a sensitive software or application based on architectural or design decisions, the CIA is a valuable activity for the decision-makers. Our survey report highlights that professionals put an extensive time to the CIA's activities, and generally give importance to their content.

In chapter-5 we proposed a tool that allows automated selection of the appropriate IA approach that a developer should follow to be GDPR compliant. This tool not only saves the time of the developer from manually selecting the approach, it also appropriate filter to get the resultant approach that is suitable for the particular case, and hence it save cost.

However the only disadvantage of this tool is that it is applicable to be used only by those companies that are not yet GDPR compliant or are into the transition phase.

CHAPTER-7

REFERENCES

[1] Bohner, S. A. (1996). A graph traceability approach for software change impact analysis.

[2] de La Vara, J. L., Borg, M., Wnuk, K., & Moonen, L. (2016). An industrial survey of safety evidence change impact analysis practice. IEEE Transactions on Software Engineering, 42(12), 1095-1117.

[3] Leveson, N. G. (2016). Engineering a safer world: Systems thinking applied to safety (p. 560). The MIT Press.

[4] Rovegård, P., Angelis, L., & Wohlin, C. (2008). An empirical study on views of importance of change impact analysis issues. IEEE Transactions on Software Engineering, 34(4), 516-530.

[5] Borg, M., Gotel, O. C., & Wnuk, K. (2013, May). Enabling traceability reuse for impact analyses: A feasibility study in a safety context. In 2013 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE) (pp. 72-78). IEEE.

[6] Bjarnason, E., Runeson, P., Borg, M., Unterkalmsteiner, M., Engström, E., Regnell, B., ... & Feldt, R. (2014). Challenges and practices in aligning requirements with verification and validation: a case study of six companies. Empirical software engineering, 19(6), 1809-1855.

[7] Nair, S., de la Vara, J. L., Sabetzadeh, M., & Falessi, D. (2015). Evidence management for compliance of critical systems with safety standards: A survey on the state of practice. Information and Software Technology, 60, 1-15.

[8] Regan, G., McCaffery, F., McDaid, K., & Flood, D. (2013, June). Investigation of traceability within a medical device organization. In International Conference on Software Process Improvement and Capability Determination (pp. 211-222). Springer, Berlin, Heidelberg.

[9] [Online]: http://serg.cs.lth.se/fileadmin/serg/ImpRec_EvalStudy/ImpRec_InterviewGuides.pdf

[10] Runeson, P., Host, M., Rainer, A., & Regnell, B. (2012). Case study research in software engineering: Guidelines and examples. John Wiley & Sons.

[11] Klevin, A. (2012). People, process and tools: A Study of Impact Analysis in a Change Process. Department of Computer Science, Faculty of Engineering, LTH, Lund University.

[12] Angerer, F., Grimmer, A., Prähofer, H., & Grünbacher, P. (2019). Change impact analysis for maintenance and evolution of variable software systems. Automated Software Engineering, 26(2), 417-461.

[13] Musco, V., Carette, A., Monperrus, M., & Preux, P. (2016, May). A learning algorithm for change impact prediction. In 2016 IEEE/ACM 5th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE) (pp. 8-14). IEEE.

[14] Abdulkhaleq, A., & Wagner, S. (2015, April). A controlled experiment for the empirical evaluation of safety analysis techniques for safety-critical software. In Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering (pp. 1-10).

[15] Knight, J. C. (2002, May). Safety critical systems: challenges and directions. In *Proceedings of the 24th international conference on software engineering* (pp. 547-550).

[16] Galbo, S. P. D. (2017). *A Survey of Impact Analysis Tools for Effective Code Evolution* (Doctoral dissertation).

[17] Lindvall, M., & Sandahl, K. (1998). How well do experienced software developers predict software change?. *Journal of Systems and Software*, *43*(1), 19-27.
[18] Planning, S. (2002). The economic impacts of inadequate infrastructure for software testing. *National Institute of Standards and Technology*.
[19] Arnold, R. S., & Bohner, S. A. (1993, September). Impact analysis-towards a framework for comparison. In *1993 Conference on Software Maintenance* (pp. 292-301). IEEE.
[20] Buckner, J., Buchta, J., Petrenko, M., & Rajlich, V. (2005, May). JRipples: A tool for program comprehension during incremental change. In *13th International Workshop on Program Comprehension (IWPC'05)* (pp. 149-152). IEEE.
[21] Jayaraman, G., Ranganath, V. P., & Hatcliff, J. (2005, April). Kaveri: Delivering the indus java program slicer to eclipse. In *International Conference on Fundamental Approaches to Software Engineering* (pp. 269-272). Springer, Berlin, Heidelberg.
[22] Zhang, L., Kim, M., & Khurshid, S. (2012, November). FaultTracer: a change impact and regression fault analysis tool for evolving Java programs. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering* (pp. 1-4).
[23] Galbo, S. P. D. (2017). *A Survey of Impact Analysis Tools for Effective Code Evolution* (Doctoral dissertation).
[24] Goeritzer, R. (2011, May). Using impact analysis in industry. In *Proceedings of the 33rd International Conference on Software Engineering* (pp. 1155-1157).
[25] de Souza, C., & Redmiles, D. (2008, May). An empirical study of software developers' management of dependencies and changes. In *2008 ACM/IEEE 30th International Conference on Software Engineering* (pp. 241-250). IEEE.
[26] Li, B., Sun, X., Leung, H., & Zhang, S. (2013). A survey of code-based change impact analysis techniques. *Software Testing, Verification and Reliability*, *23*(8), 613-646.
[27] Tóth, G., Hegedűs, P., Beszédes, Á., Gyimóthy, T., & Jász, J. (2010, September). Comparison of different impact analysis methods and programmer's opinion: an empirical study. In *Proceedings of the 8th International Conference on the Principles and Practice of Programming in Java* (pp. 109-118).
[28] Stallinger, F., Neumann, R., Schossleitner, R., & Zeilinger, R. (2011, May). Linking software life cycle activities with product strategy and economics: Extending ISO/IEC 12207 with product management best practices. In *International Conference on Software Process Improvement and Capability Determination* (pp. 157-168). Springer, Berlin, Heidelberg.
[29] O'Connor, R. V., & Laporte, C. Y. (2014). An innovative approach to the development of an international software process lifecycle standard for very small entities. *International Journal of Information Technologies and Systems Approach (IJITSA)*, *7*(1), 1-22.
[30] Aydan, U., Yilmaz, M., Clarke, P. M., & O'Connor, R. V. (2017). Teaching ISO/IEC 12207 software lifecycle processes: A serious game approach. *Computer Standards & Interfaces*, *54*, 129-138.
[31] Rajlich, V. (2014). Software evolution and maintenance. In *Proceedings of the on Future of Software Engineering* (pp. 133-144).
[32] Xu, B., Qian, J., Zhang, X., Wu, Z., & Chen, L. (2005). A brief survey of program slicing. *ACM SIGSOFT Software Engineering Notes*, *30*(2), 1-36.
[33] Harman, M., Binkley, D., Gallagher, K., Gold, N., & Krinke, J. (2009). Dependence clusters in source code. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, *32*(1), 1-33.
[34] Black, S. (2001). Computing ripple effect for software maintenance. *Journal of software maintenance and evolution: research and practice*, *13*(4), 263-279.

[35] Lehnert, S. (2011, September). A taxonomy for software change impact analysis. In *Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution* (pp. 41-50).
[36] Zanjani, M. B., Swartzendruber, G., & Kagdi, H. (2014, May). Impact analysis of change requests on source code based on interaction and commit histories. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (pp. 162-171).
[37] Hattori, L., Guerrero, D., Figueiredo, J., Brunet, J., & Damásio, J. (2008, May). On the precision and accuracy of impact analysis techniques. In *Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)* (pp. 513-518). IEEE.
[38] Wilkerson, J. W. (2012, September). A software change impact analysis taxonomy. In *2012 28th IEEE International Conference on Software Maintenance (ICSM)* (pp. 625-628). IEEE.
[39] Han, J. (1997, July). Supporting impact analysis and change propagation in software engineering environments. In *Proceedings Eighth IEEE International Workshop on Software Technology and Engineering Practice incorporating Computer Aided Software Engineering* (pp. 172-182). IEEE.
[40] Yan, J., Sarukkai, S., & Mehra, P. (1995). Performance measurement, visualization and modeling of parallel and distributed programs using the AIMS toolkit. *Software: Practice and Experience*, *25*(4), 429-461.
[41] Sahakyan, A. (2019). Software change impact analysis with respect to data protection.
[42] Jones, C. (1986). How not to measure programming quality. *Computerworld*.
[43] Kitchenham, B. A., Travassos, G. H., Von Mayrhauser, A., Niessink, F., Schneidewind, N. F., Singer, J., ... & Yang, H. (1999). Towards an ontology of software maintenance. *Journal of Software Maintenance: Research and Practice*, *11*(6), 365-389.
[44] Kilpinen, M. S. (2008). *The emergence of change at the systems engineering and software design interface* (Doctoral dissertation, University of Cambridge).