A Systematic Study of Advancements in Change Impact Analysis Techniques

Ankit Dhamija, Sunil Sikka

Abstract*: Introducing changes in the software during development and post development is a very frequent activity. Reasons to changes includes client's changing requirements, fixing left over bugs and other security issues, adding some new functionality and so on. Implementing the suggested changes in software may bring adverse effects which may degrade its quality or introduce new bugs which in turn may increase the software maintenance cost. Therefore a systematic change management process is required. A systematic process for managing software change is already in place i.e. Change Impact Analysis (CIA). It analyzes the nature of each change request and tries to incorporate those changes in the software by following a stepwise procedure. A number of innovative CIA techniques have been proposed in the literature. The current paper conducts a systematic study of recent developments, techniques and tools in the area of CIA and highlights the future research scope in this area.

***Index Terms*: Change Impact Analysis, Dependency, Feature Location, Traceability.**

I. INTRODUCTION

Software Maintenance is the most costly process in the whole life cycle of Software Development [1] as a lot of time, effort and money goes into code understanding and analysis. It is necessary to ensure that the software remains reliable even after several modifications that may arise due to changing clients' requirements, migration to new platform, introducing a new feature or functionality. Every such modification or change needs to be carefully analyzed and the consistency of the software with other interacting entities must be ensured. Change Impact Analysis (CIA) is a systematic process under Software Maintenance that includes variety of techniques through which the possible effects a change may have on other software elements can be ascertained. Bohner et al. [2] defined CIA as "a process of identifying the potential consequences of a change, or estimate what needs to be modified to accomplish a change". Before them, Pfleeger et al. [3] defined CIA as "the evaluation of the many risks associated with the change, including estimates of the effects on resources, effort, and the schedule". Before that, Horowitz et al. [4] defined CIA as "an examination of an impact to determine its parts or elements". The most widely used CIA process [5] starts with an analysis of change request and the creation of Change Set that includes all the initial impacted areas due to change introduction.

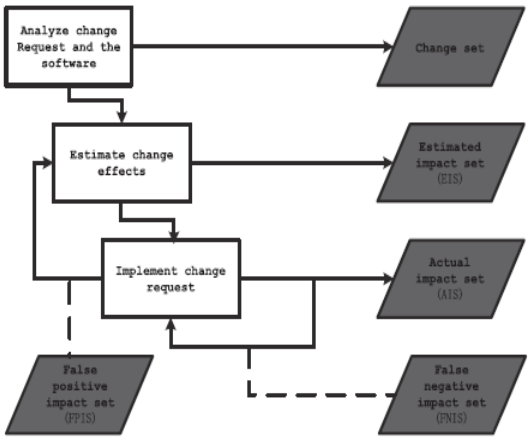

Figure 1: CIA Process

[42] presented a detailed analysis of the various techniques of identifying the initial location under CIA.

Then, Estimated Impact Set (EIS) is created impact of the changes on other elements in the software are estimated using various change impact analysis techniques. Then, Actual Impact Set (AIS) is created that includes all the locations where changes have been implemented. Two more sets False Negative Impact Set (FNIS) and False Positive Impact Set (FPIS) are also created that denotes under estimation and over estimation of impacts respectively. The ultimate objective of carrying out CIA is to make sure that the difference between EIS and AIS is zero. The relationship among all these sets can be represented as:

$$\mathbf{(EIS+FNIS)-FPIS=AIS} \quad (1)$$

The objective of the whole CIA process is about generating an Estimated Impact Set that is equal to the Actual Impact Set but it's seldom achieved. This objective can be achieved only through a careful selection of apt CIA techniques. To check the accuracy of CIA techniques, various metrics exist; however, the most accepted are Precision and Recall [6] where Precision refers to the extent to which estimated impacts (EIS) coincide with the actual impacts brought up by changes; *Recall* is to what extent the Estimated Impact Set covers the actual changes.

$$\mathbf{Precision= |EIS \cap AIS| / |EIS|} \quad (2)$$

$$\mathbf{Recall= | EIS \cap AIS | / |AIS|} \quad (3)$$

EIS with high precision is an indicator of less time spent in identifying the location of changes and implementing those changes. EIS with high recall means all the impacts of those proposed changes will be taken into consideration. Broadly, CIA techniques can be categorized into three types: Traceability CIA, Dependence Based CIA and Experiential Based CIA.. [41] presented a detailed study on the various

Revised Manuscript Received June 07, 2019

Ankit Dhamija, Amity School of Engineering & Technology, Amity University Haryana, Gurgaon, India.
Sunil Sikka, Amity School of Engineering & Technology, Amity University Haryana, Gurgaon, India.

other software metrics for performing CIA.

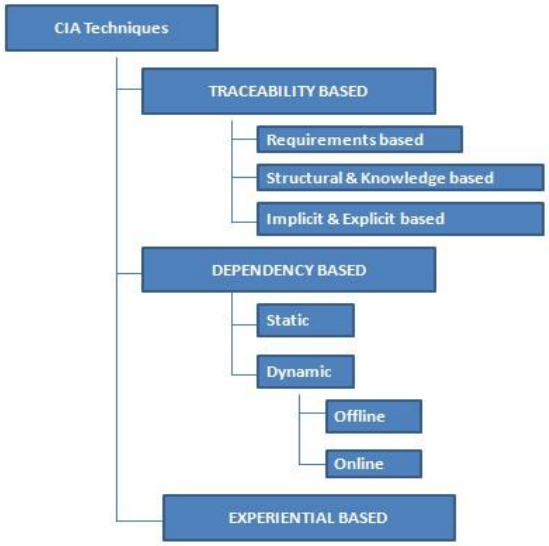

Figure 2: Types of CIA

Traceability Based CIA: Under Traceability CIA, relationships between various software elements like requirements, design, code, documents, test cases etc are identified and analyzed to ascertain possible effect of introducing a software change. It attempts to discover linkages between various software elements.

Dependency Based CIA: Under these, CIA is performed by taking into consideration various software artifacts like variables, logic, modules etc and analyzing their interdependence to find out the effects of initiating a change. While Traceability based CIA performs impact analysis at different levels of abstraction, Dependency based CIA performs it at the same level i.e. requirements to requirements, design to design and code to code. The requirements and design based approaches rely on UML Diagrams and use case diagrams for performing impact analysis which are not reliable and not that mature. But source code based approaches are being proposed by researchers in recent past. These code based approaches can either be static or dynamic. The static techniques work when the program is not in execution. These are performed through syntax & semantic analysis, textual analysis, program's change history repositories. Dynamic CIA includes offline and online CIA. It works by recording and analyzing the data gathered while the program is in execution, like data gathered from execution traces. A CIA technique where information gathered is analyzed after the execution of program has finished comes under offline CIA and information retrieved when the program is still in execution comes under Online CIA.

Experiential CIA: In this type, informal methods like Review meeting protocols, discussion within teams, and individual expertise and judgments are followed for finding out the results of a change and modification [9]. However, from research perspective and importance levels, the first two types (Traceability and Dependency based CIA) are the most relevant. Various studies presented relevant and consequential review on CIA techniques [7, 8, 9]. The last survey paper on CIA was from 2012 and during last 7 years, a lot of advanced techniques have been proposed in literature.

In this study, a comprehensive analysis of the current developments in CIA is presented where CIA techniques are categorized according to different CIA aspects.

The remaining paper is structured as follows: Section 2 presents the research methodology and review process. Section 3 presents the detailed analysis of various CIA types and techniques proposed; in Section 4, a comprehensive analysis of the recent tool support for CIA is presented; Section 5 present the findings & outcome of the paper and in Section 6, the conclusion of the paper is presented.

II. RESEARCH METHODOLOGY AND REVIEW PROCESS

Following research questions (RQ) have been formulated for conduct the review in a systematic manner in order to obtain the recent research status in CIA.

RQ1. What are the recent advancements in various types of Traceability based CIA techniques and Dependency based CIA techniques?

RQ2. What are the latest tools for performing CIA?

RQ3. What is the future scope of research in CIA with respect to CIA types?

To achieve the above research objectives, following process was followed:

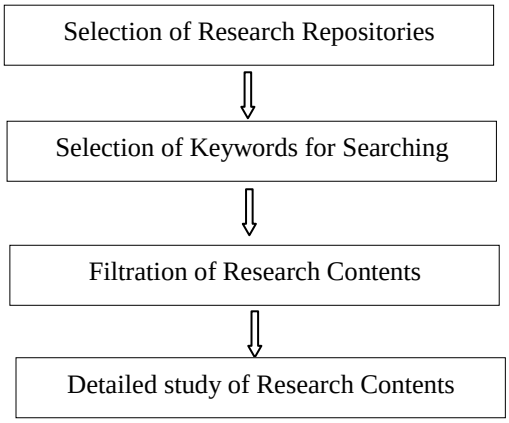

Figure 3: Stepwise Review Process

Research papers from IEEE Computer Society Digital Library (http://ieeexplore.ieee.org), ACM Digital Library (http://portal.acm.org), Elsevier (http://www.sciencedirect.com), Springer (http://link.springer.com, http://springeropen.com/journals and Others (Google Scholar, ResearchGate, Academia.edu) were searched using relevant keywords. The search results are presented in Table 1.

Table 1: Search Results

Research Repository	"Change Impact Analysis" + "Traceability"	"Change Impact Analysis" + "Dynamic"	"Change Impact Analysis" + "Static"	Total
IEEEXplore	33	31	31	95
ACM Digital Library	15	14	14	43
Springer	140	176	141	457

Elsevier	68	74	64	206
Total	256	295	250	**801**

A total of 801 research papers were found using the keywords. Thereafter, the process of paper filtration was performed where the paper relevancy on the basis of Paper Title was carried out, which further reduced the search results to 250 papers. Based on the detailed study of the abstract, 33 papers were taken into consideration.

III. TRACEABILITY BASED CHANGE IMPACT ANALYSIS TECHNIQUES

Lucia et al. [7] defined traceability as "the ability to describe and follow the life of an artifact, in both a forwards and backwards direction". This means that when a document like requirement, use case etc, that are associated with the feature that requires some change, traceability assists in locating areas in code and design that is required to be retained. The authors categorized papers related to traceability as exploratory, experimental and empirical & case based.

A. Exploratory Studies

SS Khan et al. [10] proposed an Aspect Oriented Requirements Engineering based approach in which concern-oriented dependency taxonomy was developed which helps in capturing the relationship dependencies between requirements level concerns and its architectural level demonstration. The output from the results depicted that the components are unstable. Gotel et al. [11] did a comprehensive analysis where they figured out the challenges and the future research directions in traceability analysis. They provided a roadmap for ubiquity in traceability research.

B. Experimental Studies

Kama et al. [12] proposed an approach for the development phase that had two stages: Developing Class Interaction predictions where requirement and design artifacts are analyzed; and Performing Impact Analysis stage which focused on identifying the impacted set of classes. During the experimental validation of their approach, they proved their technique's accuracy over existing ones. Shahid et al. [13] proposed a tool called Hybrid Coverage Analysis Tool (HYCAT) to manage the traceability before and after introducing changes in software artifacts. The experimentation of the tool was performed on On-board Automobile (OBA) and their results showed promising and remarkable output when compared with existing approaches. Kugele et al. [14] proposed an algorithm based on model methodology to facilitate trace link visualization and understand the artifact importance and the impact an artifact has on the other one. Dit [15] proposed the use of Genetic Algorithms in IR approaches for recovering traceability link. Their approach IR-GA finds out the near optimal solution that is used at every step of the Information Retrieval process. Kchaou et al. [16] proposed a graph based technique for modeling the structural dependencies and an IR technique for managing the semantic traceability among use case documentation and sequence diagrams. They performed a quantitative experimentation of LSI frequency and Inverse Document Frequency on JHotDraw 7.4.1 and their results showed a much better value of precision and recall using LSI.

Huang et al. [17] gave a modeling based approach for fully utilizing the benefits of traceability metrics and for planning and executing traceability strategies in a graphical modeling environment. They used graphs for capturing strategic traceability decisions and XML for representing model elements. Nejati et al. [18] proposed an approach based on Systems Modeling Language (SysML) where their modeling approach identifies the effect of introducing changes in requirements on design. They applied static slicing technique for fetching the approximated set of model elements that are impacted and thereafter did a ranking of the resulting set elements for predicting the impact of the elements. Their results revealed that 4.8% of the entire design needs to be inspected for identifying the elements those is actually-impacted.

Diaz et al. [19] proposed a technique targeting PLA's depending on traceability and rule based inference engine that account for variability. Their approach has provisions for specifying variability in PLA's, tracing variability between requirements and PLA's and documenting PLA's knowledge. Javed et al. [20] considered a hypothesis that an architecture supported by traceability links will prove to be more efficient and they tested it by designing two experiments for finding out the traceability links impact on retrieved assets quality and quantity during evolution analysis of the architecture. Their results showed that it can significantly reduce the missing and incorrect assets quantity.

C. Empirical and Case Study Based

Espinoza et al. [21] proposed three features called "user-defined traceability links, roles and linkage rules" to be supported by traceability models. They proposed a traceability metamodel (TmM) that included these three features that provided a higher degree of automation and more extensive support to agile process stakeholders. A methodical trace acquisition and maintenance process will be possible using their approach. Hayes et al. [22] proposed tool called "RETRO" (REquirements TRacing Ontarget) that traces requirements to automate the generation of requirements tracing matrix (RTMs). Their results showed that more correct links are found using RETRO and it takes 33% less time taken by manual tracing.

For UML 2.0 class diagrams, Briand et al. [23] proposed a traceability approach in which CIA activity was categorized into horizontal and vertical CIA. They stated that UML models should remain in a consistent state when changes are being performed. Their approach worked by formalizing changes to those models, introducing refinements and traceability links related to those refinements using the Operations Control Language (OCL).

Ghabi et al. [24] stated that time is saved and quality is improved when traceability is maintained between software artifacts but it is not captured at the most appropriate time. So a language for traceability capturing was proposed by them that allow engineers to express their hypothesis about the traceability relationship involving various program elements and code that may not be consistent or complete. They also demonstrated that their approach is correct and scalable. Almasri et al. [25] proposed a model-based approach on telecommunication or embedded systems where their model used dependencies and generates

two impact sets and addresses "extended finite-state machine (EFSM) models". An empirical evaluation over six EFSM models was performed to find out its usability. Their results revealed that introduction of a single change bring an impact between 14 to 38 % of the total model size. Table 3 summarizes the Traceability based CIA Techniques:

Table 2: Summary of Recent work in Traceability based CIA

Year/ Publisher	Reference	Type of Paper/ Study	Concepts covered	Technique Discussed /Tools Proposed	Results
2008, IEEEXplore	Lucia et al. [7]	Review Paper	Traceability Recovery, Link Evolution.	Information Retrieval based, Heuristic based, Data Mining Based approaches	Distinguished between three types of Traceability Links: Vertical and Horizontal, Structural and Knowledge based, Implicit and Explicit.
2009, IEEEXplore	Khan et al. [10]	Exploratory	Aspect Oriented Requirements Engineering.	Concern-oriented dependency Taxonomy. Analyzed the captured traces to find concern overlap, intertwine, and conform.	Intertwining is useful to identify unstable architectural components.
2012, IEEEXplore	Gotel et al. [11]	Exploratory	Discussion on two projects: Center of Excellence for Software Traceability (CoEST) and Grand Challenge of Traceability(GCT) [36]	Traceability Information Models, Automated trace creation & maintenance, traceability economics	Provided a Traceability roadmap and Presented the challenges and Proposed Research Directions in Traceability.
2012, IEEE Computer Society	Kama et al. [12]	Experimental and Case study based	Proposed a combined CIA approach for the development phase. Called Software Development Phase Change Impact Analysis (SDP-CIA)	Static techniques like Use Case Maps (UCM), class interaction prediction with Impact Prediction Filters (CIP-IPF) technique and requirement interdependency. Dynamic techniques like Influence Mechanism, Path Impact	Better accuracy with SDP-CIA than CIP-IPF.
2016, IEEEXplore	Shahid et al. [13]	Experimental and Case study based	Proposed an approach and tool to manage impact analysis and traceability before and after a change implementation	Hybrid Coverage Analysis Tool (HYCAT) applied on On-Board Automobile (OBM).	Experimentation results show in numbers, the impact of requirements on methods, classes, packages and test cases, providing accurate and efficient results. Also during Feature Analysis, HYCAT got maximum mean value than GRAYzer and JavaCodeCoverage tool.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

2016, IEEE Computer Society	Dit [15]	Experimental	Usage of Genetic Algorithms for configuring and assembling an Information Retrieval process. (IR-GA approach)	Text and Search based Software Engineering, Latent Dirichlet Allocation-Genetic Algorithm (LDA-GA)	Provides the near optimal solution.
2016, IET Journal, IEEE	Kchaou et al. [16]	Quantative and Experimental	Usage of graph technique to model the structural dependencies and an information retrieval.	Traceability model, Trace link Visualization	No visual clutter with dense traceability graph.
2009, IEEEXplore	Huang et al. [17]	Experimental	strategic traceability decisions, modeling reusable trace queries using sequence diagrams.	Traceability Model representation, Trace query definition as sequence diagram, composite trace queries.	Implemented a prototype using XML documents and associated queries using a tool named BASEX.
2016, ACM	Nejati et al. [18]	Experimental, Industrial Case Study	Technique of automatically identifying the effect of changes made to requirements on design.	Systems Modeling Language (SysML) based approach using Natural Language Processing.	Results state that only 4.8% of the entire design is required to be inspected for identifying the actually-impacted elements.
2011, Springer	Espinoza et al. [21]	Case Study based	Proposed three features that create a Traceability Metamodel for supporting agile methods more effectively.	Traceability metamodel (TmM), Agile methods, Test Driven Development (TDD), Storytest Driven Development(SDD)	Results depict higher degree of automation and more extensive support to agile processes.
2011, Springer	Diaz et al. [19]	Experimental	Proposed new technique focusing on Product Line Architecture (PLA).	Traceability-based method and rule based inference engine, flexible PLA Metamodel, PLAK Metamodel for documenting PLA knowledge, latent semantic analysis (LSA)	The inference engine is at theoretical design stage and yet to be implemented.
2015, Springer	Javed et al. [20]	Experimental, Empirical	Testing of hypothesis that traceability links can be quite helpful in doing impact analysis of changes in software architecture.	Method based on hypothesis.	Increase in quality of architecture, reduction in number of missing/ incorrect assets.
2016, Springer	Almasri et al. [25]	Empirical, Experimental	state based systems using model dependencies	Study conducted on six extended finite-state machine (EFSM) models, Model dependencies and modifications.	Results stated that model density and data density may affect the sizes of impact sets. Also, limited scope impact sets are generated when compared with the whole model.
2007, Springer	Hayes et al. [22]	Case Study based	Generation of requirements traceability matrix (RTM)	RETRO tool.	Accurate results.

2009, Elsevier	Briand et al. [23]	Experimental and Case Study Based	Approach based on formalization of original refinements and traceability links through the use of OCL.	VIATool (Vertical Impact Analysis Tool)	Correct and complete change taxonomies & refinement rules. .
2015, Elsevier	Ghabi et al. [24]	Experimental and Case Study Based	Unreliable Traceability documentation and its maintenance	A language for capturing traceability is proposed. TraceAnalyzer tool is proposed	All uncertainties resolved.

IV. DEPENDENCY BASED CHANGE IMPACT ANALYSIS TECHNIQUES

Dependency based CIA works by taking into consideration various software artifacts like variables, logic, modules etc and analyzing their interdependence to find out the effects of initiating a change. These may be static or dynamic or both.

A. Static Techniques

Static techniques work by analyzing the software artifacts by not executing the program but through syntax & semantic analysis, textual analysis, program's change history repositories. These methods focus on Program Structure.

M Sherriff et al. [26] proposed the usage of singular value decomposition (SVD) to find out the impact of introducing an alteration through software change records analysis. Through their methodology, file clusters are generated that tend to change together earlier. Their approach was also compared with *PathImpact* and *CoverageImpact* techniques. Results showed comparable outcomes and identified other files that may also be impacted. Sharma & Suryanarayana [27] stated that most CIA approaches at present lack support for hidden dependencies and inter-granular change impact queries. They introduced a static automated tool called *Augur* for code analysis, which takes care of these limitations by maintaining semantic and environment dependencies between source code entities across granularities. Quantitative evaluation was conducted on open source and industrial projects where quite good precision and recall were recorded. T Rolfsnes et al. [28] proposed the use of evolutionary coupling through a new algorithm called "Targeted Association Rule Mining for All Queries" (TARMAQ). They compared it with *ROSE* tool and *SVD* tool and found that it is better than the two and best suited to perform robust CIA for heterogeneous systems. Musco et al. [29] used four types of call graphs to propose a technique to forecast impact circulation. Ten Open source JAVA Projects and five mutation operators were used for creating 17000 mutants to understand how errors propagate. Their results revealed that best tradeoff between precision and recall is given by the most basic call graph.

B. Dynamic Techniques

It includes offline and *online CIA.* It is performed when program is in execution. A CIA technique where information gathered is analyzed after the execution of program has finished comes under *Offline CIA* and information retrieved when the program is still in execution comes under *Online CIA.*

Cai and Santelices [30] proposed a framework with three instances that generate very precise impact sets in a cost effective manner. They used static dependencies and execution traces to bring great precision value. In another paper [31], they proposed a dynamic technique for Sensitivity Analysis called *SENSA* that generated statement level impact sets. They empirically evaluated SENSA on open source JAVA projects and case studies. Cai and Thain [32] proposed tool called DISTIA which estimated the impacts spread inside and outside the process boundaries by partially ordering distributed method-execution events and exploiting message-passing semantics. Their results revealed that the analysis gets finished within one minute and also reduced the size of impact set by 43%. Rajan and Kroening [33] defined metric that quantify the change impact using two program versions to predict behavioral impact. Their approach is unique as analysis is run on both the versions of the program. They also evaluated their metric on three case studies. Cai and Santelics [34] performed an analysis of predictive accuracy of dynamic CIA using a two- way process. Their approach used execution differencing & sensitivity analysis to find out the precision and recall. Their outcome revealed that most techniques performing a cost effective dynamic analysis gives inaccurate results with 38–50% average precision and 50–56% average recall in most cases.

Table 6 summarizes the above mentioned static and dynamic approaches.

Table 3: Summary of Recent work in Dependency based CIA

Year, Publisher & Author	Paper Overview	Type of CIA	Technique Discussed	Results
2015 IEEE M Sherriff et al. [26]	Usage of singular value decomposition to find out the impact of introducing a change	Static	Singular value decomposition (SVD)	Compared with *PathImpact & CoverageImpact and achieved* 60% reduction in developer effort in static analysis.
2016 IEEE Computer Society Sharma & Suryanarayana	Proposed Augur which takes care of hidden dependencies and inter-granular change impact queries.	Static	Change Impact Query language, Semantic and Environment dependency	Results indicate average precision of 55% and average recall of 85%.

[27]				
2016 IEEE Computer Society T Rolfsness et al. [28]	Proposed a new algorithm called TARMAQ for evolutionary coupling mining.	Static	Evolutionary Coupling, Association Rule Mining	Empirically evaluated on the two industrial and four open source systems. TARMAQ gives better results than ROSE and SVD and runs faster than SVD. Significant (p-value < 0.00001)
2015 IEEE Cai and Santelices [30]	Designed and evaluated DDIA framework and its three new instances.	Dynamic	execution traces, dynamic points to data and statement coverage	Compared with the baseline technique PI/EAS, IAPRO is found to be constantly much (160%–200%) more precise.
2015 IEEE Cai and Santelices [31]	Proposed a tool called SENSA (Sensitivity analysis and execution differencing)	Dynamic	Dynamic forward slicing	More effective than slicing techniques Takes 6.57% less effort than static slicing, 13.49% less than dynamic slicing.
2016 ACM Cai & Thain [32]	Proposed *DISTIA Tool* for dynamic analysis of distributed systems	Dynamic	message passing semantics	Reduces the size of impact set by 43% with run time overhead less than 8%. Avg Precision of around 70%.
2015 Springer Rajan & Kroening [33]	Measures change impact using multiple program versions.	Dynamic	Program Dependency Graphs (PDG's), Control Flow Graphs (CFGs)	The metric was successful in capturing impact for deletions as both program versions were used in the analysis.
2015 Springer Cai & Santelics [34]	To forecast the accuracy of dynamic CIA	Dynamic	Sensitivity Analysis and Execution Differencing	Results concluded that most approaches provide Inaccurate data. Average precision: 38–50% Average recall: 50–56%.
2016 HAL Musco et al. [29]	Mutation Testing, Impact Propagation, Impact Prediction	Static	Call Graph Based	Better results when call graphs is used and good execution time.

V. TOOL SUPPORT FOR CIA

The following tools supporting CIA have been proposed by researchers in recent times:

a. TRIC [35] [36]: Tool for Requirements Inferencing and Consistency checking (TRIC) works on software requirements using formal requirement semantics to perform CIA and requirements predictions. In another paper [36], the authors enhanced the software's functionality and made provisions to include the features like display of inconsistent proposed changes, proposing, propagating and implementing and predicting changes and their impact in the requirements model.

b. ImpactMiner [37]: The tool estimates an impact set using a textual analysis and dynamic tracing, history mining and querying SVN Repositories, evolutionary item set mining techniques. It is used as a plugin to the Eclipse tool and has a very intuitive GUI where two tabs- Feature view and Results view give the user a clear understanding of the results.

c. SafeRefactorImpact [38]: SafeRefactorImpact is a tool for evaluating whether a transformation saves the program actions based on change impact analysis. It works by analyzing changes applied on Java or AspectJ programs, and generating test cases for the impacted methods. It uses Safira, the change impact analyzer that recognizes the methods impacted

d. CodeDiff [39]: It generates an estimated impact set (EIS) by taking a textual change request and releasing a code snippet that is indexed using Latent Semantic Indexing technique.

e. Cobra [40]: The Cobra tool is used to analyze the projects written primarily in C, C++, and Java.

f. TraceAnalyzer [24]: Implemented as an Eclipse plug-in, the tool has provision for different input views. It keeps the traditional trace matrix(TM) and list of imputes on the right and left sides respectively. Also, it has features that help engineers in finding out the foot print graph, highlighting problems related to correctness and granularity, and detaching them.

Table 4 enlists the availability of these and other recently proposed tools for carrying out Change Impact Analysis.

Table 4: Recent Tool Support in CIA

Tool Name	Availability
Chianti	Free, integrated with Eclipse http://eclipse.org
ImpactMiner	Freely available http://www.cs.wm.edu/semeru/ImpactMiner/
Impala	Not available freely
Jripples	Freely available http://jripples.sourceforge.net/

JD edwrds	https://docs.oracle.com/cd/E17984_01/doc.898/e14719/impactanalysistool.htm
TRIC	http://trese.cs.utwente.nl/tric/pse.org
Codediff	Not available freely
Cobra	Open source, available at https://software.nasa.gov/software/NPO-50050-1
TraceAnalyzer	Not available freely

VI. FINDINGS AND FUTURE SCOPE

This paper provides an updated analysis on the recent developments in CIA. The critical analysis of CIA on the basis of research questions framed in section 2 yields following findings:

a. Findings reveal that the traceability analysis and dependency based analysis are the most popular approaches in CIA where most of the work in traceability based CIA are based on Information Retrieval approaches. The major challenges in traceability lies in finding trace links, traceability information models and automated trace creation and maintenance. The recent work under dependency based CIA takes care of the efficiency of their approach and compares it with other approaches on the most important parameters like Precision, Recall and F Value.

b. Tools like SENSA, DISTIA, TARMAQ, IAPRO, and AUGUR have been proposed which work well on certain types of inputs. The most popular techniques among researchers remain as graph based techniques, dependency based, method execution and execution traces. However, the approaches like Singular Value Decomposition (SVD), Sensitivity analysis, Execution differencing etc are some of the new and innovative techniques that are being proposed in recent times which have given considerably good results.

c. The recently proposed CIA techniques are following an interdisciplinary approach where techniques are proposed using areas like Data Mining that includes Associan Rule mining, item set mining, Matural Language Processing, Information retrieval, Graph based techniques and so on.

d. There is a lot of scope of research in CIA in terms of finding hidden dependencies. There are dependencies which are clearly visible from a class's interface but some dependencies remain hidden. Techniques may be proposed to find out such hidden dependencies among software artifacts like requirements, design and code.

e. The new areas of research may include CIA techniques based on fuzzy logic model where a sense of fuzziness is brought upon by the researchers in their proposed solution, which is able to predict the percentage or chances of introduction of a change in the system, based on some parameters.

f. Then, there is a scope of research about inter-granularity change. Then, the techniques that can prove to be improving on their results every time they are used will be very much useful to perform CIA effectively; thus the self improving techniques should be focused upon.

VII. CONCLUSION

In this research paper, the authors have attempted to review the various CIA techniques where a study was carried out by framing systematic research questions which were answered through systematic analysis. The various findings under each research question was critically analyzed and presented. Various techniques for feature location, traceability analysis and dependency analysis have been discussed and their comparison has been presented. The paper also presented a detailed analysis of the recent tools available for performing CIA. Also, the gap areas have been suggested that need to be focused upon for further research. The authors feel that the detailed analysis of the recent work in CIA will enable the researchers working in this area to choose the most appropriate tool and technique for enhancing the existing work and for proposing the novel techniques.

REFERENCES

1. Li W, Henry S. Maintenance support for object-oriented programs. *Journal of Software Maintenance: Research and Practice* 1995; **7**(2):131–147.
2. Bohner, S.A. and Arnold, R.S. 1996. An introduction to software change impact analysis. Software Change Impact Analysis, IEEE Comp. Soc. Press, pp. 1–26, June 1996.
3. Pfleeger, S.L. Bohner, S.A. 1990. A framework for software maintenance metrics. Proceedings of the International Conference on Software Maintenance, Washington, DC, 1990; 320–327.
4. Horowitz, E. Williamson, R.C. 1986. SODOS: a software documentation support environment—its definition. IEEE Transactions on Software Engineering, vol 12(8) pp 849–859.
5. Bohner, S.A. 2002. Software change impacts—an evolving perspective. Proceedings of the International Conference on Software Maintenance, Montréal, Canada, pp 263–272.
6. Hattori, L. Guerrero, D. Figueiredo, J. Brunet, J. and Damsio, J. 2008 On the precision and accuracy of impact analysis techniques. Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science, Portland, Oregon, pp 513–518.
7. De-Lucia, A. Fasano, F. Oliveto, R. 2008. Traceability management for impact analysis. Proceedings of the International Conference on Software Maintenance, Beijing, China,21–30.
8. Sun, X. Li, B. Wen,W. 2015. Static change impact analysis techniques: A comparative study, The Journal of Systems and Software 109 (2015) pp 137–149.
9. B. Li, X. Sun, H. Leung, and S. Zhang. A survey of code-based change impact analysis techniques. Software Testing, Verification and Reliability, 23:613–646, 2013.
10. S. S. Khan and S. Lock, "Concern tracing and change impact analysis: An exploratory study," in Proceedings of the 2009 ICSE Workshop on Aspect-Oriented Requirements Engineering and Architecture Design, Vancouver, BC, Canada, May 2009, pp. 44–48
11. O Gotel and A. Egyed, The quest for Ubiquity: A roadmap for software and systems traceability research, Proceedings of the 2012 IEEE 20th International Requirements Engineering Conference (RE), p.71-80, September 24-28, 2012.
12. N. Kama, F. Azli, A change impact analysis approach for the software development phase, Proceedings - Asia-Pacific Software Engineering Conference, APSEC, Volume: 1 Pages: 583-592 Published: 2012
13. M Shahid and S Ibrahim, Change impact analysis with a software traceability approach to support software maintenance, 13th International Bhurban Conference on Applied Sciences and Technology (IBCAST), 12-16 Jan 2016.
14. S Kugele and Dl Antkowiak, Visualization of Trace Links and Change Impact Analysis, 2016 IEEE 24th International Requirements Engineering Conference Workshops, 12-16 Sept 2016.
15. B Dit, Configuring and Assembling Information

Retrieval based Solutions for Software Engineering Tasks, 2016 IEEE International Conference on Software Maintenance and Evolution, pp 641-647.
16. D Kchaou, N Bouassida, H B Abdallah, UML models change impact analysis using a text similarity technique, IET Journals, IET Software, Volume: 11, Issue: 1, 2 2017, pp 27-37.
17. J C-Huang, J H Hayes2, J. M. Domel, Model Based Traceability, 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering 2009 pp 6-10.
18. S Nejati, M Sabetzadeh, C Arora, L C. Briand, Automated Change Impact Analysis between SysML Models of Requirements and Design, FSE'16, November 13–18, 2016, Seattle, WA, USA, pp 242-254.
19. Díaz, J., Pérez, J., Garbajosa, J., Wolf, A. L.: Change Impact Analysis in Product-Line Architecture. Proceedings of the 5th European Conference on Software Architecture (ECSA 2011), Essen, Germany, September 13-16, 2011.
20. M A Javed and U Zdun, The Supportive Effect of Traceability Links in Change Impact Analysis for Evolving Architectures – Two Controlled Experiments, ICSR 2015, LNCS 8919, pp. 139–155, 2014. Springer International Publishing Switzerland 2014.
21. A Espinoza and J Garbajosa, A study to support agile methods more effectively through traceability, Innovations Syst Softw Eng (2011) SPRINGER 7:53–69.
22. J. H. Hayes A. Dekhtyar S. Sundaram A. Holbrook S. Vadlamudi and A. April "Requirements tracing on target (retro): Improving software maintenance through traceability recovery " Innovations in Systems and Software Engineering: A NASA Journal vol. 3 no. 3 pp. 193-202 Sep 2007.
23. L C. Briand, Y Labiche , Tao Yue, Automated traceability analysis for UML model refinements, Information and Software Technology 51 (2009) 512–527.
24. A Ghabi, A Egyed, Exploiting traceability uncertainty among artifacts and code, The Journal of System and Software, 2015, pp 178-192.
25. N Almasri, L Tahat, B Korel, "Toward automatically quantifying the impact of a change in systems", Software Quality Journal, SPRINGER published on 09 May 2016.
26. M Sheriff and L Williams, Empirical Software Change Impact Analysis using Singular Value Decomposition, 1st International Conference on Software Testing, Verification, and Validation, 2008
27. T Sharma and G Suryanarayana, Augur: Incorporating Hidden Dependencies and Variable Granularity in Change Impact Analysis, 2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation, pp 73-78.
28. T Rolfsnes, Stefano Di Alesio, R Behjati, L Moonen and D W. Binkley, Generalizing the Analysis of Evolutionary Coupling for Software Change Impact Analysis, 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering, pp 201-212.
29. Vincenzo Musco, Martin Monperrus, Philippe Preux. A Large-scale Study of Call Graph-based Impact Prediction using Mutation Testing. Software Quality Journal, Springer Verlag, 2016.
30. H Cai and R Santelices, A Framework for Cost-Effective Dependence-Based Dynamic Impact Analysis, 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER), pp 231-240.
31. H Cai, R Santelices and S Jiang, Prioritizing Change Impact Analysis via Semantic Program-Dependence Quantification, IEEE Transactions on Reliability,2016, Volume: 65, Issue: 3 pp1114 – 1132
32. H Cai and D Thain, DistIA: A Cost-Effective Dynamic Impact Analysis for Distributed Programs, 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), 2016, pp 344-355.
33. A Rajan and D Kroening, Measuring Change Impact on Program Behaviour, Validation of Evolving Software, Springer International Publishing Switzerland 2015, pp 125-145.
34. H Cai, R Santelices, "A comprehensive study of the predictive accuracy of dynamic change-impact analysis", The Journal of Systems and Software 103 (2015) 248–265.
35. A Goknil R van Domburg, I Kurtev, K van den Berg, F Wijnhoven, Experimental Evaluation of a Tool for Change Impact Prediction in Requirements Models: Design, Results and Lessons Learned, IEEE 4th International Model-Driven Requirements Engineering Workshop (MoDRE), 2014.
36. A Goknil, I Kurtev, K van den Berg, K Spijkerman, Change impact analysis for requirements: A metamodeling approach, Information and Software Technology 56 (2014) 950–972.
37. Dit, B. Wagner, M. Wen, S. Wang, W. Vásquez, M.L. Poshyvanyk, D and Kagdi, H. 2014. ImpactMiner: A Tool for Change Impact Analysis. ICSE'14.
38. Mongiovia, M. Gheyi, R Soaresa, G. Teixeira, L. and Borba, P. 2014. Making refactoring safer through impact analysis. Science of Computer Programming pp 39–64.
39. Gethers, M. Dit, B. Kagdi, H and Poshyvanyk, D. 2012, Integrated impact analysis for managing software changes, Proceedings of the 2012 International Conference on Software Engineering, June 02-09, 2012, Zurich, Switzerland.
40. Holzmann, G.J. 2016. "Cobra: A Light-Weight Tool for Static and Dynamic Program Analysis", Innovations in Systems and Software Eng., pp. 1-15.
41. A Dhamija, S Sikka, , "Software Change Management: a Quantified Perspective", International Journal of Engineering & Technology [Online], Volume 7 Number 3.12 , July 2018.
42. A Dhamija, S Sikka, "*A Systematic Review of Feature Location Techniques under Software Change Impact Analysis*", International Journal of Computer Sciences and Engineering, Vol.7, Issue.3, pp.184-192, 2019.

AUTHORS PROFILE

Mr. Ankit Dhamija pursued Master of Computer Applications from Kurukshetra University Kurukshetra in 2008. He is currently pursuing Ph.D. and currently working as Assistant Professor in Amity Business School, Amity University Haryana since 2012. He has published more than 15 research papers in reputed international journals including Scopus indexed journals and presented several research papers at International conferences of repute. His main research work focuses on Software Maintenance, Cloud Security and Privacy, Management Information System. He has 10 years of teaching experience and 4 years of Research Experience.

Dr Sunil Sikka completed his Ph.D from Maharishi Dayanand University, Rohtak. and currently working as Associate Professor in Amity School of Engineering & Technology at Amity University Haryana since 2014. He has published more than 20 research papers in reputed international journals including Thomson Reuters (SCI & Web of Science) and conferences including IEEE and it's also available online. His main research work focuses on Software Engineering, Software Testing, Data Mining, Internet of Things and Computational Intelligence based education. He has 12 years of teaching experience and 8 years of Research Experience.