TEST Engineering & Management

November-December 2019
ISSN: 0193-4120 Page No. 4435 - 4445

A Novel Process using Impact Analysis over Requirement Phase with Impact Transition Model

Sathyarajasekaran K
(Corresponding Author)
sathyarajasekaran.k@vit.ac.in

Ganesan R
Vellore Institute of Technology,
School of Computer Science and Engineering,
Chennai Campus
ganesan.r@vit.ac.in

Article Info
Volume 81
Page Number: 4435 - 4445
Publication Issue:
November-December 2019

Article History
Article Received: *5 March 2019*
Revised: *18 May 2019*
Accepted: *24 September 2019*
Publication: *23 December 2019*

Abstract:
Impacts are the key aspects in software engineering when it identified earlier can reduce the overhead in all the phases of development. Although the existing works are addressed on change impact analysis over software development phases. We believe that analyst experience during requirement elicitation can provide way to identify the impact based requirements. In this paper author discuss a new process for impact analysis over requirements. We address how informal requirements of the system under impact analysis to evolve and represented using different models like transition and sequence. The proposed approach is based on the impact-transition structure that was elicit impact oriented requirements and its analysis. This work illustrated through a case study to realize the proposed approach with precision, recall and F-measure.

1. Introduction

The requirements are the one that explained as the needs of users, customers or the market, which consequently administrate the development tasks like architecture, designing, implementing and examining. Requirements are further defined as a well-known and completely understood earlier design and analysis in the utopian point of view in software development. On considering the real-world scenario, the product attained a shape by understanding, insights and general known process. The users or customers can change their mind set in accordance with their needs, as the underlying needs after requirements might get challenging to grasp and may have overly optimistic time plans. Because of these factors, the environment cannot set the requirements as fixed. Hence, the common developmental work can get disturbed as because of the unexpected alterations at times.

2. Change Impact Analysis

The impact analysis has been approached based on the two usual identified models. The main focus of Traceability based IA is concentrated on sketching the dependencies among artifacts such as design documents, requirements, and source code files. The main concentration of Dependence based IA is on analyzing the detail effect of ripple or change in the software system initialized using software change.

November-December 2019
ISSN: 0193-4120 Page No. 4435 - 4445

Bohner&Arnold[3,4,5,6,7,8] stated that, while changing the software in accordance to an existing system or requirement of feature, it will typically affect the multiple files which explain the resources, database tables, classes, configuration files, and other files. This considered file types can termed as Software Life Cycle Objects (SLOs).

SLO which is as well named as working products, or software products, acts as the center of impact analysis. The artifact that created at the time of a project, like a class, a requirement, an architectural component, and so on is termed as SLO. SLOs are associated with each other via a web of relations. Relations may be among SLOs of the similar kind, and among SLOs of diverse kinds. For instance, two requirements have been interrelated for signifying the relationship among each other. A requirement has been as well associated with the architectural module, for instance, for signifying that the component executes the requirement.

There may begin a requirement for change within multiple parts of the software, when SLOs can pose dependencies among one another. This process started with the assigning of impacted SLOs. Initially, the first procedure assessment has offered a SLOs set, which gets directly impacted by the change and is called as Starting Impact Set (SIS). Some other terms used in papers and literature, i.e. Rajlich called that as initial impact set.

Though, there are typically interactions and dependencies in SLOs explained within SIS to erstwhile SLOs which were not involved in SIS. On investigating the interactions and dependencies, the identification of SLOs has been made. Estimated Impact Set (EIS) is defined as the SLO's dependant set computed to impact by the change on SIS. Those can be further termed as secondary modifications. The starting impact set is also included within this estimated impact set.

The Actual Impact Set (AIS) is formed by the affected modules, when performing the software change to SLOs. The impact analysis is analyzed to be operated fltlessly, when EIS is assigned equivalent to AIS. To be noted, as there are multiple techniques on performing the software change, AIS is not generally considered as unique.

In the existed IA approaches survey, the author determines the False Negative Impact Set (FNIS) and False Positive Impact Set (FPIS). The FPIS is involved with SLO's that has determined within EIS, still not incorporated within AIS. These modules do not change as per the estimation. The FNIS is the contrast process of FPIS that involved with SLO's that has determined within EIS, still are incorporated within AIS. These module changes still not determined with the phase of estimation.

3. Change Impact Analysis Techniques

The associated reviews on change impact analysis [18,19, 20] in accordance to the used techniques on impact analysis are explained in this section. In this, the major concern is on the core approaches that introduced in the literature, investigate their basic notations, and implement them with regards to their probability of supporting multiperspective impact analysis. The three major needs on analyzing the existed approach on change impact analysis were taken as research objectives and these objectives are explained below.

- The ability on analyzing heterogeneous kinds of software artifacts.
- The support for developers that tried on comprehending the impacts of their changes.
- The support for diverse kinds of change operations.

The following are the discussion about the six crucial terms of change impact analysis techniques.

November-December 2019
ISSN: 0193-4120 Page No. 4435 - 4445

3.1Traceability Analysis

On considering the traceability Impact Analysis [31, 32, 33, 34], the capturing of specifications, tests, requirements, and design elements are made, and these correlations have been analyzed for determining the capacity of an initiating change. The manual determination of what and who gets affected by the change is considered as error-prone and time consuming over the critical projects having thousands of artifacts. By using the impact analysis, the peoples and items that are impacted can be automatically enlightened at the time of occurrence of change.

Information Retrieval

In the literature, several techniques for IR-based traceability [22, 23] detection were implemented, like latent semantic indexing or vector space model. The names and identifiers of software artifacts are analyzed by this model as same as the IR-based change impact analysis techniques. Therefore, the creation of traceability relation among both is made, when the two artifacts names are "similar". In order to enhance the precision of the link detection, more techniques are used further with pre-processing approaches, like stop word elimination or word stemming. Numerous approaches are analyzed and concluded that the merging of diverse approach can attain the precise outcomes. Though, IR-based techniques are lacked in needed precision and recall while comparing over other techniques and are hence not fit for dependency detection in multiperspective circumstances, because of the detection of too many false-positives. Moreover, the determination of detected relations type is not made by IR-based approaches because of its incapability, which directs to the restriction of detected dependencies' reusability for short impact analysis. Table 1.1 delineates the synopsis on IR approaches for Multiperspective dependency detection

Mining of Software Repositories

As same as the discussion in the previous approaches on history-based impact analysis, MSR-based [24, 25, 26] techniques has been as well applied for traceability recovery. Hence, the similar hypothesis is subjugated for change impact and that is defined as: there is an existence of traceability relation among them, when two artifacts were regularly changed together. Therefore, the same limitations are applied for traceability detection by MSR. Initially, the detection of evolutionary couplings is carried out, while neglecting the other types of dependencies entirely. In the second, diverse kinds of software artifacts are generally developed in various repositories and therefore will not share a general "history" which has been extracted. Further, if version history is assigned as missing or incomplete in the early stages of software development as well when the software is in an unstable state, the MSR has not been applied. Table 1.2 represents the synopsis of MSR approaches for

Dependency Detection Rules

In order to detect the dependencies among software artifacts, the developers and researchers have defined and executed a set of rules and record them under traceability links. For accomplishing this task, the rules can query the attributes, structure or relations of software artifacts for determining the dependencies. Typical instances are found over the requirements traceability fields and in model-driven engineering works. These rule-based techniques for traceability detection have shared the similar merits and demerits as the rule-based approaches for change impact analysis. In contrast, more consistent outcomes have been provided and permitted for a superior understanding of the outcomes while comparing with other techniques. Moreover, this method is as well capable of determining the detected dependency relation's type. Similarly, the detection rules concept is simpler to acclimatize with any other software

November-December 2019
ISSN: 0193-4120 Page No. 4435 - 4445

artifacts, as narrative rules were formed for them. While considering the other techniques, it may need rigorous changes in their original algorithms. The synopsis on rule-based approaches [27, 28] for Multiperspective dependency detection is demonstrated.

Semantic Wikis and Ontologies

The semantic modelling concept [29] is deployed for the fourth group of traceability detection techniques, in order to elicit the dependency relations among software artifacts. The fact that motivates these techniques are by the software development that contains numerous stakeholders, everyone uses the own individual vocabulary. Whereby presents the similar concepts of synonyms and homonyms, and the predictable drift among software artifacts because of the ongoing evolution. These proposed techniques have indexed the software documents with an ontology which has permitted to retrieve the knowledge from the software artifacts from where the traceability links has been gathered. The main limitations and difficulty of such techniques is on the primary creation and concept definition, which comprised of software and the modelling within ontology. On contrary, this method permits the traceability detection in the heterogeneous framework and probably conflicting software artifacts.

Machine Learning

Machine learning techniques [29, 30] is considered to be the algorithm which is capable of automatically "learning" the traceability links from software artifacts on the basis of a provided dependency relation's training set. Either the developers or the comprised combination of manually evolved links has supplied these training sets and based on the granularity levels, the links are identified by program analysis and run time monitoring. These approaches have the ability to eliciting such links when the training sets include the traceability links that connect the heterogeneous software artifacts. Furthermore, when the reflection of types is made using the training set, these techniques can be probably capable on differentiating the diverse kinds among the relations. Thus, the case studies have depicted that the precision of the gained outcomes gets highly varies.

4. Impact Analysis Process over Requirements

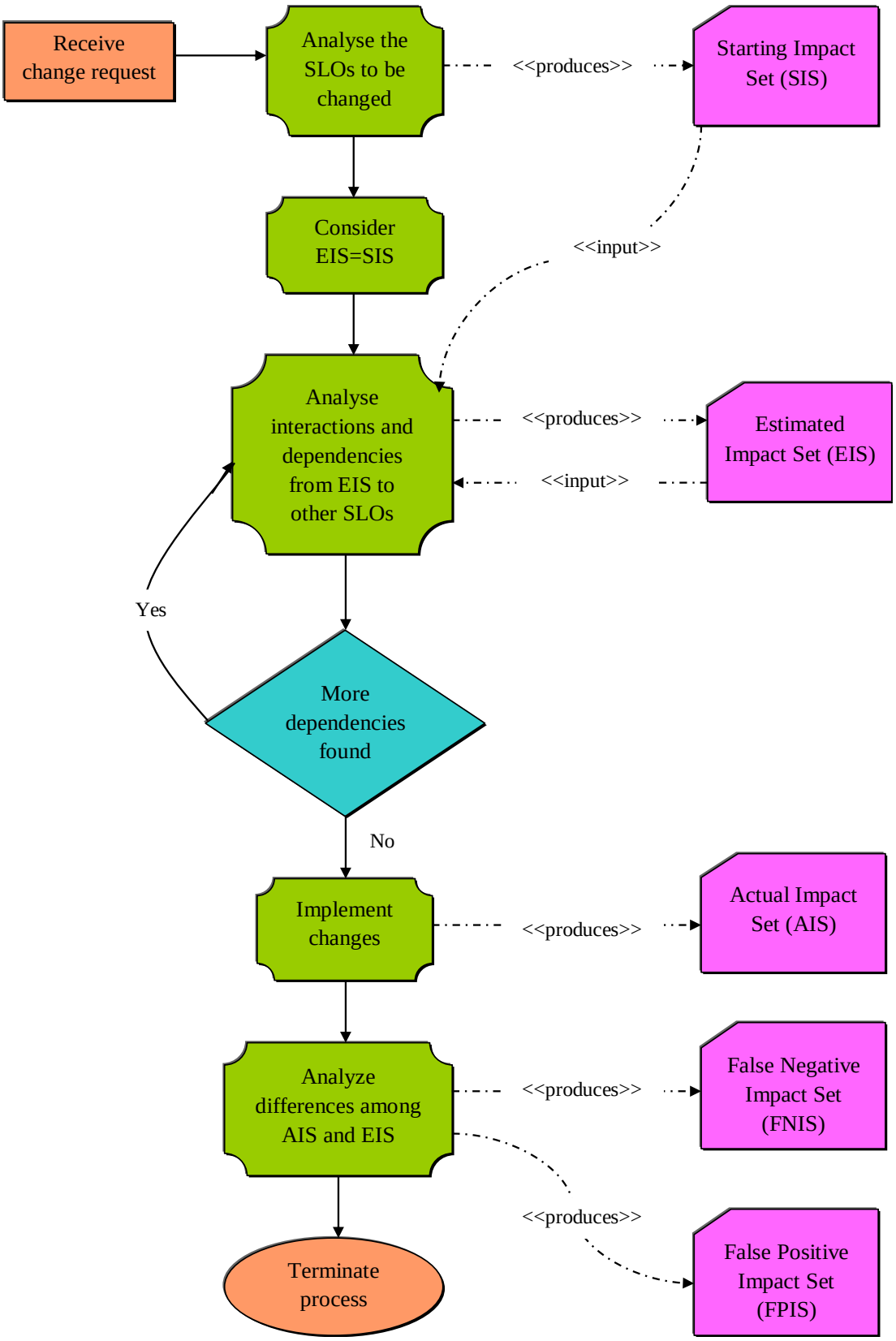
The change impact analysis [35, 36] process can be either formal or informal. The change locations are identified in an informal way by the developers with no iteration procedure and are exclusively on the basis of the developer's expertise and their system knowledge close by. The analysis can be made not under the basis of formal addressing of impact analysis.

One of the formal ways is on performing the impact analysis in a recursive and incremental manner. Fig. 1 symbolizes the art on the impact analysis process.

1 Generally, on the basis of developer expertise, the SIS is determined and analyzed for a change request, as the same as a manual process.
2 The execution of impact analysis is happened after the identification of SIS, for discovering the interactions and dependencies among the Software Lifecycle Objects (SLOs) incorporated in SIS and other SLOs. The EIS is formed by this set of SLOs. Further, the analysis of interactions and dependencies of SLOs incorporated in EIS are made and summed up to EIS, thereby after every iterative analysis, the enlarging of EIS is made.
3 The changes are made to the system. The AIS is formed by the SLO's set that was really impacted.
4 The SLOs which has been changed and can be involved in AIS, still that hadn't within EIS develops the FNIS.
5 The SLOs which has not been changed and cannot involve within AIS, still that has been contained in EIS develops the FPIS.

November-December 2019
ISSN: 0193-4120 Page No. 4435 - 4445

6 When EIS = AIS, the process of change impact analysis was assumed to be faultless and needs previous determination on the entire changes.



The SLOs are assigned with one of these following marks:

November-December 2019
ISSN: 0193-4120 Page No. 4435 - 4445

- ✓ Blank, mean the SLO hasn't examined nor scheduled for inspection
- ✓ Changed, denotes the SLO can get affected as per the change and is a branch of EIS
- ✓ Unchanged, indicates the SLO as examined and the result of the analysis is defined with the non-impact of the changes over SLO
- ✓ Subsequently, schedules the SLO for examining at the time of impact analysis
- ✓ Propagating, indicates the SLO has not get affected in accordance with the change directly, still there may be a change in the dependencies of this SLO, i.e. the change can get proliferated to dependant SLOs

At first, the SLOs can be assigned as blank. After that begins the impact analysis process and the identification of SIS is made as per the Fig. 1. The files that contained within SIS are assumed as changed. Subsequently, the entire dependent SLOs are assumed as next that determines the first EIS. Afterward, the procedure of analysis process goes on continuously by means of monitoring the EIS SLOs and their appropriate dependencies are marked with suitable marking. The propagating SLO dependencies are marked as next, when using the propagating mark. This process goes on iteratively till there contains no SLOs with marking as next has been found.

4.1Interaction and Dependencies with Impact Transition Model

An impact transition modelgiven in fig.1.2 shows the third section in fig. 1.1 (Analyse interaction and dependencies). In this, each noderepresent possible estimated impact object over a change or set of estimated impact objects sequence. Model impact (MI) can be an edge from node P to Q represents requirements impact or a set of sequential impact represented by P, likely transit to node Y which also represents same.

Here, in figure 1.2 nodes are represented as I_A, I_BI_D, I_C, I_E, I_FI_G , MI can be calculated based on the impact weight assigned by the experts over different SLOs.

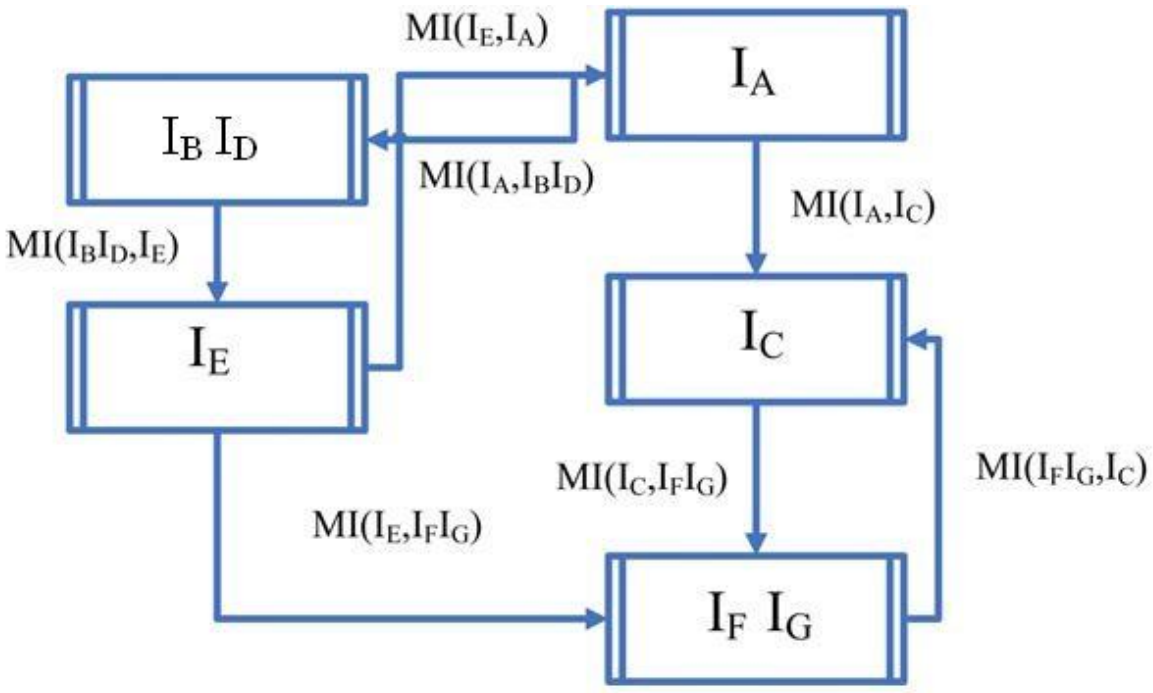


Figure1.2. Example Impact Transition Model for Interaction and Dependencies

4.2 Evaluation

For the evaluation of the impact transition model, this paper uses precision, recall and F-Measure. True Positive (TP) represents a sample that is positive and predictively positive, False Positive (FP) represents a sample that is actually negative but positively predicted, False Negative (FN) represents a sample that is actually positive but negatively predicted, and True Negative (TN) represents a sample that is actually negative and predicted to be negative, available tables 4.1 indicates.

Table 4.1 Prediction Matrix Table

Prediction matrix		Prediction	
		positive	negative
Actual	positive	TP	FN
	negative	FP	TN

the above evaluation indicators are calculated based on the application considered.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \quad (3)$$

Precision is used to evaluate the system's ability to reject erroneous samples in samples, Recall is used to indicate the ability to find the right sample, and F-Measure is the harmonic mean of Precision and Recall, which is a comprehensive evaluation of the classification. The above three indicators are all the overall evaluation of the classification,

4.3 A Case Study

In this work we consider a student portal case study to show the proposed transition model with the adopted process. A student portal which handles 25000 students of different streams in a university.

- All students can able to receive the exam schedule but few of them not able to get the schedule in their student portal.
- Set of students not able to take exam scheduled. Since, out of 25K set of students exam schedule is not reflected in their student portal. This is an serious problem relates to reliability and quality

Impact Analysis over the above scenario has to be done well in advance during the requirement change or new requirement occurs. Similarly, other set of scenarios can be considered. Like way can be considered for three applications with five different modules using change impact transition (CIT) and no change impact transition (NonCIT). Following figure shows that the details.

November-December 2019
ISSN: 0193-4120 Page No. 4435 - 4445

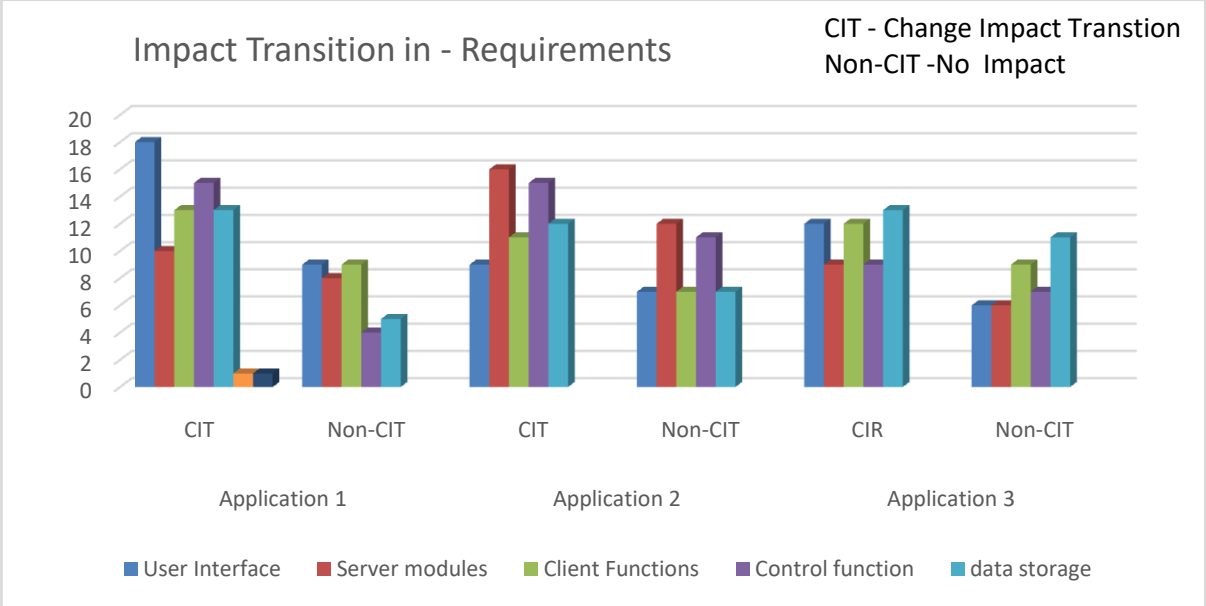


Figure 4.1 Impact transition over different applications with requirements

Every now and again, there is an opposite relationship among precision and recall, where it is possible to increase one at the cost of reducing the other. Software application with impact and non-impact requirements provides an illustrative example of the tradeoff. Consider ansoftware analyst tasked with classifies impact requirementsfrom an application over Non-impact requirements. The system analyst needs to avoid all of the non-impact requirements since any remaining may become impact or non-impact requirement. Conversely, the analyst must not remove impact requirements since that would leave the application with weak process. This decision increases recall but reduces precision. On the other hand, the analyst may be more traditional in the applicationthat analyst ensures that he removes only irrelevant requirements only.This decision increases precision but reduces recall. That is to say, greater recall increases the chances of removing relevant requirements (negative outcome) and increases the chances of removing all cancer cells (positive outcome). Greater precision decreases the chances of removing impact requirements (positive outcome) but also decreases the chances of removing irrelevant requirements (negative outcome).

Typically, precision and recall scores are not talked about in confinement. Rather, either values for one measure are analyzed for a fixed level at the other measure (for example exactness at a review level of 0.75) or both are consolidated into a solitary measure. Measures that are a blend of precision and recall are the F-measure (the weighted consonant mean of exactness and review). Here, we concentrated dependent on the analyst who chose the impact and non-impact requirements dependent on three distinctive application he/she has worked. Here, below figure shows the precision, recall and F measure.

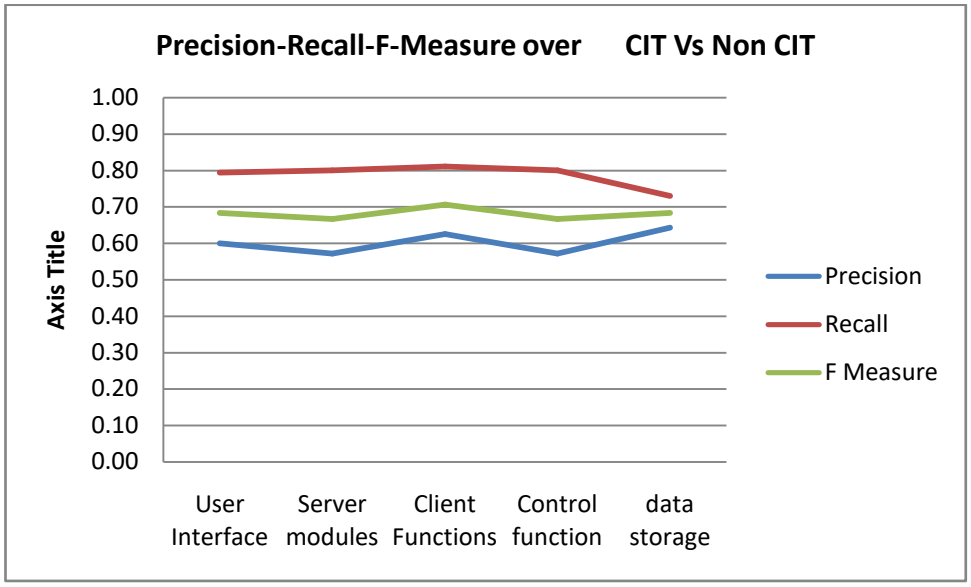


Figure 5.2 Precision-Recall and F measure in impact transition

6. Conclusions and Future works

There is a huge requirement for systems to find the impact of changes in early stages. Assume that if it is identified when we process the initial stage of requirements. This work is trying to address the structure of the transition model and process steps for impact analysis. This analysis carried over different system and produced the transition model to identify the priority of work.

This work can be extended to learning of requirements and predicts what might be the feature impacts. There are other enhancement over the system in different capacity it can be worked as an future enhancement.

7. References

1. Robert Heinrich, Sandro Koch, Suhyun Cha, Kiana Busch, Birgit Vogel-Heuser, “Architecture-based change impact analysis in cross-disciplinary automated production systems”, Journal of Systems and Software, vol. 146, pp. 167-185, December 2018.
2. Ines Hajri, ArdaGoknil, Lionel C. Briand, Thierry Stephany, “Change impact analysis for evolving configuration decisions in product line use case models”, Journal of Systems and Software, vol. 139, pp. 211-237, May 2018.
3. S. A. Bohner, “Impact analysis in the software change process: a year 2000 perspective,” in Proceedings of the 12th International Conference on Software Maintenance (ICSM’96), Monterey, CA, pp. 42–51, November 1996.
4. S.A. Bohner, “Extending software change impact analysis into COTS components”, in: 27th Annual NASA Goddard Software Engineering Workshop, pp. 175–182, 2002.
5. S.A. Bohner, D. Gracanin, “Software impact analysis in a virtual environment”, 28th Annual NASA Goddard Software Engineering Workshop, pp. 143–151, 2003.
6. Robert S. Arnold and Shawn A. Bohner, “Impact analysis - towards a framework for comparison”, In Proceedings of the IEEE Conference on Software Maintenance (CSM ’93), pages 292–301, Montreal, Quebec, Canada, September 1993.
7. S.A. Bohner, Software change impacts – an evolving, perspective, ICSM’02, 2002, pp. 263 271.
8. S. A. Bohner and R. S. Arnold, “Software Change Impact Analysis”, Los Alamitos, CA, USA: IEEE Computer Society Publications Tutorial Series, 1996.
9. BixinLi, XiaobingSun, HaretonLeung," Combining concept lattice with call graph for impact analysis", Advances in Engineering Software, vol.53, pp.1-13, November 2012

November-December 2019
ISSN: 0193-4120 Page No. 4435 - 4445

10. BasCornelissen, AndyZaidman, DannyHolten, LeonMoonen, ArievanDeursena, JarkeJ.vanWijk," Execution trace analysis through massive sequence and circular bundle views", Journal of Systems and Software, vol.81, no.12, pp.2252-2268, December 2008
11. P. Wu, J. Wang and B. Tian, "Software Homology Detection With Software Motifs Based on Function-Call Graph," IEEE Access, vol. 6, pp. 19007-19017, 2018.
12. B. G. Ryder, "Constructing the Call Graph of a Program," IEEE Transactions on Software Engineering, vol. SE-5, no. 3, pp. 216-226, May 1979.
13. S. Casale-Brunet and M. Mattavelli, "Execution Trace Graph of Dataflow Process Networks," IEEE Transactions on Multi-Scale Computing Systems, vol. 4, no. 3, pp. 340-354, 1 July-Sept. 2018.
14. F. Lanubile and G. Visaggio, "Extracting reusable functions by flow graph based program slicing," in IEEE Transactions on Software Engineering, vol. 23, no. 4, pp. 246-259, April 1997.
15. B. Korel, "Computation of dynamic program slices for unstructured programs," IEEE Transactions on Software Engineering, vol. 23, no. 1, pp. 17-34, Jan. 1997.
16. M. Mock, D. C. Atkinson, C. Chambers and S. J. Eggers, "Program slicing with dynamic points-to sets," IEEE Transactions on Software Engineering, vol. 31, no. 8, pp. 657-678, Aug. 2005.
17. K. B. Gallagher and J. R. Lyle, "Using program slicing in software maintenance," IEEE Transactions on Software Engineering, vol. 17, no. 8, pp. 751-761, Aug. 1991.
18. X. Sun, B. Li, C. Tao, W. Wen and S. Zhang, "Change Impact Analysis Based on a Taxonomy of Change Types," 2010 IEEE 34th Annual Computer Software and Applications Conference, Seoul, pp. 373-382, 2010.
19. J. W. Wilkerson, "A software change impact analysis taxonomy," 2012 28th IEEE International Conference on Software Maintenance (ICSM), Trento, 2012, pp. 625-628.
20. M. Ceccarelli, L. Cerulo, G. Canfora and M. Di Penta, "An eclectic approach for change impact analysis," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, pp. 163-166, 2010.
21. Bohner Shawn A., "Impact Analysis in Software Change Process: A Year 2000 Perspective", IEEE, pp. 42 - 51, 1996.
22. MaurizioPighin, GiorgioBrajnik," A formative evaluation of information retrieval techniques applied to software catalogues", Journal of Systems and Software, vol.52, no.2–3, pp.131-138, 1 June 2000.
23. AndrásKicsi, ViktorCsuvik, LászlóVidács, FerencHorváth, ÁrpádBeszédes, TiborGyimóthy, FerencKocsis," Feature analysis using information retrieval, community detection and structural analysis methods in product line adoption", Journal of Systems and Software, vol.155, pp.70-90, September 2019.
24. XiaobingSun, BixinLi, HaretonLeung, BinLi, YunLi," MSR4SM: Using topic models to effectively mining software repositories for software maintenance tasks", Information and Software Technology, vol.66, pp.1-12, October 2015.
25. Meng Yan, Ying Fu, Xiaohong Zhang, Dan Yang, Jeffrey D. Kymer, "Automatically classifying software changes via discriminative topic model: Supporting multi-category and cross-project", Journal of Systems and Software, vol.113, pp.296-308, March 2016.
26. OlivierVandecruys, DavidMartens, BartBaesens, ChristopheMues, ManuDe Backer, RafHaesen," Mining software repositories for comprehensible software fault prediction models", Journal of Systems and Software, vol.81, no.5, pp.823-839, May 2008.
27. MeghaBhushan, ShivaniGoel, KaramjitKaur," Analyzing inconsistencies in software product lines using an ontological rule-based approach", Journal of Systems and Software, vol.137, pp.605-617, March 2018.
28. DongWang, RuibingHao, DavidLee," Fault detection in Rule-based Software systems", Information and Software Technology, vol.45, no.12, pp.865-871, 15 September 2003.
29. OmarMeqdadi, NouhAlhindawi, JamalAlsakran, AhmadSaifan, HatimMigdadi," Mining software repositories for adaptive change commits using machine learning techniques", Information and

November-December 2019
ISSN: 0193-4120 Page No. 4435 - 4445

Software Technology, vol.109, pp.80-91, May 2019.
30. QinbaoSong, XiaoyanZhu, GuangtaoWang, HeliSun, HeJiang, ChenhaoXue, BaowenXu, WeiSong," A machine learning based software process model recommendation method", Journal of Systems and Software, vol.118, pp.85-100, August 2016.
31. CatiaTrubiani, AchrafGhabi, AlexanderEgyed," Exploiting traceability uncertainty between software architectural models and extra-functional results", Journal of Systems and Software, vol.125, pp.15-34, March 2017.
32. GilbertRegan, FergalMcCaffery, KevinMcDaid, DerekFlood," Medical device standards' requirements for traceability during the software development lifecycle and implementation of a traceability assessment model", Computer Standards & Interfaces, vol.36, no.1, pp.3-9, November 2013.
33. SalomeMaro, Jan-PhilippSteghöfer, MiroslawStaron," Software traceability in the automotive domain: Challenges and solutions", Journal of Systems and Software, vol.141, pp.85-110, July 2018.
34. GabrieleBavota, AndreaDe Lucia, RoccoOliveto, GenoveffaTortora," Enhancing software artefact traceability recovery processes with link count information", Information and Software Technology, vol.56, no.2, pp.163-182, February 2014.
35. A. Oliveira Filho, "Change impact analysis from business rules," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, 2010, pp. 353-354.
36. M. Shahid and S. Ibrahim, "Change impact analysis with a software traceability approach to support software maintenance," 2016 13th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, pp. 391-396, 2016.