Analysis of Software Change Impact through Class Diagram

Himanshu Srivastava[1] Dharmendra Kumar[2]
[2]Associate Professor
[1,2]Department of Computer Science
[1,2]United College of Engineering and Research Allahabad, India

Abstract— During the entire software lifecycle requirement changes are occurred, these regular and uncontrolled require-ment change will lead to a wastage of time and effort. Impact analysis is then defined as the process of identifying the potential consequences (side effects) of a change, and estimating what needs to be modified to accomplish a change. Without proper analysis of changes which are implemented to software, they may often cause unexpected ripple effects. There are various approaches to analyze the change on existing system. Model driven approach is one of them. Impact Analysis is performed on Unified Modeling Language (UML) diagrams e.g. class diagram, use case diagram, sequence diagram. Software Change Impact Analysis needs to be computed at every change request for software systems, as developers will need fast access to the impact information for several critical software engineering tasks such as risk analysis, effort estimation, and regression testing. This paper introduces Class diagram based Approach to Analyze Software Change Impact. Proposal of this paper is a UML model based approach strictly to use Class diagram for impact analysis that can be applied before any implementation of the change, thus allowing for early decision making and change planning.

Key words: Requirement Change, Impact Analysis

I. INTRODUCTION

Requirement of users are increasing day by day. To fulfill users requirements and to compete in market, software companies need to enhance their products performance and functionality before let it being obsolete in market. So Software change management is essential activity to enrich software age.

Software Change Impact Analysis(SCIA) is an essential activity before implementing change in software. It allows to measure the effort required implementing a change and its ripple effects, impact analysis suggests those software artifacts which may be changed, and helps to identify test cases which should be re-executed to ensure that the change was implemented correctly [1]. Software Change Impact Analysis also enables developers and project leaders to determining alternate solutions without having to implement them [2].

II. SOFTWARE CHANGE IMPACT ANALYSIS

Bohner and Arnold [3] define change impact analysis as the process of identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change. Several authors address change impact analysis in the context of requirements modeling. Common techniques used to implement change impact analysis are based on either traceability or dependency relationships between the software artifacts. Traceability based impact analysis techniques work on analyzing the relationships between requirements and other development artifacts (such as design, implementation and test cases) to determine the scope of the anticipated changes, dependency-based impact analysis techniques work on a more detailed level by an-alyzing the relationships between the artifacts of the same development phase. Lehnart et al. [4] proposes impact anal-ysis techniques classification based on evaluation artifacts.

A. Code based Software Change Impact Analysis

Code based approaches investigate the impacts of changes by reasoning about inheritance relations, method call behaviour, and other dependencies between source code entities. Source code programs, class packages, classes, methods, statements, and variables are analysed to predict the propagation of changes. However, such techniques are not applicable in the early stage of software design and requirements analysis, when no source code is available. Their source code dependent nature also limits their application to programmers and project leaders. Static code analysis extracts facts from source code to build call graphs, slices, and other representations which are used to assess the impacts of a change.

B. Model based Software Change Impact Analysis

Model based analysis provides facility of SCIA for software models to keep their quality and correctness at early stage. Models, such as UML diagrams, enable the assessment of architectural changes on a more abstract level than source code. This enables SCIA in earlier stages of development and in MDD, which has become more important in recent years. But dependent on the underlying modeling language, even model based analysis provides effective impact analysis results, for example when analyzing detailed UML class diagrams.

C. Miscellaneous Artifacts based Software Change Impact Analysis

Miscellaneous artifact based SCIA is combined form of source code and model based techniques including some other software artifacts. Different researchers have used different artifacts for impact analysis e.g. some using code and SRS both and some UML models and source code both.

D. Contributing Factors for Change Request

Factors that can inflict changes to requirements during both initial developments as well as in software evolution are: 1. The objective of the system is supposed to frequently change, for example for economic or technological reasons. 2. The users change their requirements about what they want the system to do, as they understand their needs better later than the beginning. This can happen because the users initially were uncertain about what they want. 3. The specifications of the system changes. For example, increases in processor speed and capacity of computers can affect the expectations of the system. 4. The new system is developed

and released for beta testing to leading users to discover new requirements.

E. Impact Analysis Procedure

Changes should be introduced in top-down manner, start-ing with the requirements. If the requirements are decom-posed and linked to other modules, it is possible to propagate the change in a controlled way. A change impact analysis consists of the following steps:

1) Problem analysis and change specification
2) Change analysis and cost estimation, which in turn consists of:
   - Check change request validity
   - Find affected artifacts
   - Propose requirements changes
   - Assess costs of change
   - Assess cost acceptability
3) Change implementation

SCIA is performed in steps 2a, 2b, and 2c, by identifying requirements and system components affected by the pro-posed change. The analysis should be expressed in terms of required effort, time, money and available resources.

F. Strategies for Impact Analysis

There are various strategies for performing SCIA, these strategies are based on some parameters which is considered during their requirements engineering process the execution phase. These strategies can be broadly classified as following:

- Automatable
- Manual

With automatable strategies, means strategies, which are based on some tools. These strategies have the ability to provide very fine-grained impact estimation in an automated manner, but these strategies require detailed infrastructure and related information of projects (Strens and Sugden, 1996). With manual strategies, means those that are best performed by human beings (not by tools). These strategies require fewer infrastructures, but may be harder in their impact estimation than the automatable ones.

1) Automated Strategies

Automatable SCIA strategies often employ algorithmic methods in order to identify change impact. For example, relationship directed graphs for requirements can be used with other graph algorithms to identify the impact of a proposed change on the system. The prerequisite for au-tomatable strategies is a structured specification of the sys-tem. By structured, means that the specification is consistent and complete, and includes some semantic information (for example, relationship between the classes etc.). These spec-ifications can be used by tools in order to perform automatic impact analysis. Requirements dependency webs and object models are examples of structured specifications. (Lu et al., 2012) employ statistical meta-analysis techniques to investi-gate the ability of OO metrics to predict change-proneness. Some automated tools of impact analysis are (Rose,2013), (Visual paradigm, 2013). (Tao et al.,2012) study investigates the role of understanding code changes during software-development process, explores engineers information needs for understanding changes and their requirements for the corresponding tool support.

Traceability and Dependency Analysis (Antoniol et al., 2002) and (Corley et al., 2011) proposes that traceability and dependency analysis both involve examining relationships among entities in the software. They differ in scope and detail level; traceability analysis is the analysis of relation-ships among all types of models, while dependency analysis is the analysis of low-level dependencies extracted from source code. By extracting dependencies from source code, it is possible to obtain call graphs, control structures, data graphs and so on (Bohner, 1995). Since source code is the most exact representation of the system (Corley, 2011), any analysis based on it can very precisely predict the impact of a change. Dependency analysis is also the most effective strategy for impact analysis available. The drawback of using source code is that it is not available until late in the project (Khalil and Dingel, 2013), which makes dependency analysis narrow in its field of application. The identification of the primary fundamental point of traceability is based on a pre-defined search strategy and a multi-step selection process (Breivold et al., 2012) When requirements traceability exists down to the source, it can, however, be very efficient to use source code dependencies in order to determine the impact of requirements changes. A drawback is that very large systems have huge amounts of source code dependencies, which make the dependency relationship difficult to use.

Program Slicing Technique Slicing (Tip, 1995) attempts to understand dependencies using different independent slices of the program. The whole program is sliced into a decom-position slice, which contains the place of the change, and the rest of the program, a complement slice. Slicing is based on data and control dependencies in the program. Changes made to the decomposition slice around the variable that the slice is based on are guaranteed not to affect the complement slice. Slicing technique limits the scope for propagation of change and makes that scope explicit. Slicing technique is also used for slicing of documents in order to account for ripple effects as a part of impact analysis. Slicing techniques can be useful in requirements engineering to isolate the impact of a requirements change to a specific part of the system. In order to provide a starting point for the slicing technique, the direct impact of the change must first be assessed.

2) Manual Strategies

Manual impact analysis strategies do not depend as heav-ily on structured specifications as compare to automatable strategies. but manual strategies are less precise in their predictions of impact. On the other hand, they may be easier to introduce in a change management process and are still commonly employed in organizations.

Design Documentation Design documentation comes in many different forms, for example as architecture dia-grams of the system, view-based architecture models, object-oriented UML diagrams, SRS of software system and so on. The quality of design documentation depends on the purpose for which it was written, the frequency with which it is updated, and the information it contains. In some industries design documentation is written early in a project. To perform impact analysis and determine how a new or changed requirement affect the system based on design documentation requires the documentation to be up-to-date and consistent with any implementation made so far. In

addi-tion, a prerequisite for using design documentation to assess direct impact is the possibility of relating requirements to design models found in the documentation. (Choi et al., 2012) propose a rule-based approach for estimating software development cost in the requirements analysis phase.

G. Model Driven Development

Model-Driven Development (MDD) is a model-based software development approach which aims at improving build time and quality of software artifacts by focusing on UML models as abstract level artifacts rather than code[5]. UML models can be defined at different levels of abstraction to represent various aspects of the system. Dependencies between software life cycle objects are becoming more complex as many software systems grow beyond a million of lines of code. So, Model-Driven Development (MDD) is helpful in such type of systems.The potential benefits of using models are significantly greater in software than in other engineering disciplines because of the potential for a seamless link between models and the systems they represent. In the context of Model Driven Development, models also used for many reasons such as, to handle immediate errors, to add new functional requirements, to enhance some quality aspects, or to adapt to a new technological or architectural environment. UML model development can be considered to be a specialization of general software development and, similarly, requires reliable and efficient techniques and tools to manage and support UML model development. UML is a standard for modeling software systems and it is extensively used in the area of model-driven development [6]. There are many UML Models e.g. class diagram, use case diagram, sequence diagram, activity diagram etc. some author considered the hierarchies between the models. Class diagram is very close to main functionality of the software system. So we focused on class diagram. We are utilizing a functionality of class diagram in Software Change Impact Analysis. A measure of distance between a changed element and potentially impacted elements is proposed by Briad et al. [7] to prioritize the results of impact analysis according to their likelihood of occurrence. This paper provides an approach for automated Software Change Impact Analysis approach for class diagram model. Pro-posed scheme uses class responsibility collaboration (CRC) written in Visual Paradigm [8] Modelling Tool. Proposed scheme finds the impacted classes with respect to change requested. This change request is in natural language, our proposed perform change impact analysis keeping mapping with change request. The results show significant improve-ment over the existing Software Change Impact Analysis techniques.

H. Information Retrieval

SCIA approach in this thesis, is based on Information Retrieval discussed in (Marcus, 2004) used to derive the information contained in Class diagram CRC, with respect to requested change. Information retrieval processes contains following steps:

- Building a corpus,
- Natural-language processing (NLP),
- Querying, and
- Estimating an impact set.

1) Building A Corpus:

To use IR on software system, a document granularity needs to be defined, so that the corpus can be build. A document contains all the text found in a contiguous section of software artifact, such as a method, class, or package. A corpus consists of all such documents (artifacts). For instance, for impact analysis, we use crc of class diagram.

2) NLP:

Once the corpus is created, it is pre-processed using NLP techniques. For source code, operators and programming language keywords are removed. Additionally, identifiers and other compound words are split (e.g., impact Analysis becomes impact and analysis). Finally, stemming is performed to reduce words to their root forms (e.g., impacted becomes impact). This is also similarly performed for change request.

3) Running A Query:

Description of incoming change request serve as an input to this technique. An example of such a query is the like abort the transaction if invalid PIN is entered more than two times.

4) Estimating An Impact Set:

Based on the requested change we find classes which are closer to the requested change, the more textually similar the method is to the change request. An information similarity is computed be-tween the change request and all the CRC of class diagram of that particular system, the results from this list constitute an estimated most probable impact set.

I. Approach and Contribution

SCIA is an essential activity of software Maintenance. We discussed some brief points about SCIA. Source code based SCIA is effective but software development is going from classic development to MDD. In MDD only few researches have carried out in SCIA but that work is still manual, there is lack of automated SCIA approaches in MDD. Our proposed approach is concerned towards the automation of SCIA in MDD environment. This dissertation provides an approach for automated SCIA approach for Class Diagram model. Proposed scheme uses Class responsibility collaboration (CRC) written in (Visual Paradigm, 2013) Modeling Tool. Proposed scheme finds the impacted classes in Class Diagram with respect to change requested. This change request is in natural language, our proposed perform change impact analysis keeping mapping with change request. In the context of MDD, a software system typically contains many inter-related models which might have interdependent in-formation. We have utilized this interdependency for SCIA. Ensuring the overall consistency and integrity of these inter-related models requires significant effort, especially during the maintenance phase of the software system when new changes are introduced. In such cases, changes in some part of the system models have to be properly propagated to all other inter-related models otherwise it may cause violations of the consistency relationships between these inter-related models. A Class Responsibility Collaborator (CRC) model (Beck & Cunningham 1989; Wilkinson 1995; Ambler 1995) is a collection of standard index cards that have been divided into three main sections, as depicted in Figure 3.1. A class represents a collection of similar objects, a responsibility is something that a class knows or does, and a collaborator is

another class that a class interacts with to fulfill its responsibilities.

III. OBJECTIVE

Objective of this paper is a UML model based approach strictly to use Class responsibility collaboration (CRC) to find the names of impacted classes from class diagram for impact analysis that can be applied before any implementation of the change, thus allowing for early decision making and change planning.

IV. RESEARCH METHODOLOGY

Following are the proposed approach steps:

1) Read SCRF: Users or Developer, who want some change in the developed system, fill SCRF as per require-ment. After getting the filled SCRF, CCT will analyze the SCRF. The important field of the SCRF i.e. the change requested field is stored in a new directory.
2) Parsing and Extraction: In this phase, change re-quested file is parsed. Parser parses the stop words like is, are, am, this, that etc., from the stop word file. The parsed keywords are stored in an output file.
3) Impacted Classes: This phase is the information retrieval phase. After parsing the SCRF, for all extracted keywords, we search the CRC directory and gives the impacted Classes with respected to each keyword, this process follows recursively run for all keywords. For each Class, respected CRC file is name is stored.
4) Check Similar Classes: In this step similar Class names are checked.
5) Delete Similar Classes: In previous step, it might be possible that, there may be some redundant Classes. In this phase these redundant Classes are removed to avoid the ambiguity.
6) Final Impacted Classes: The final outcome of this step is the name of impacted Classes.

V. IMPLEMENTATION DETAILS

The proposed approach has implemented in three phases: To gather the change request from the user or developer, we need SCRF, thus in first phase a SCRF is designed in python language. This SCRF form contains many fields like priority, Type of change, Document affected etc. User or developer gives his change request in natural language. In second phase, we have written a NLP parser in python programming language. This NLP parses the stop words (i.e. in computing, stop words are words which are filtered out prior to, or after, processing of natural language data (text) (Wikipedia. Plagiarism, 2015). There is not one definite list of stop words which all tools used. Stop word list are updates as per CCT decisions. We have managed to parse more than 600 stop words. These stop words will growing, based on decisions taken by the CCT. After parsing change request we got some keywords as per desired change requested. To validate our approach we have taken ten examples, these examples flow of events are written in visual paradigm trial version software (Visual Paradigm, 2013). In third phase

Project Name	Change Request	Total Classes	Impacted Classes
1. Address Book	Search is also allowed by Phone number	Update Search Delete Clear Exit Save	Update Search
2. Airline Booking	Traveler my hand over his ticket to another person to travel in place of him	Login CancelTicket OnlineInquiry InternationalFlight MakePayment DomesticFlight FlightInformation BookTicket UpdateDatabase ReceiveMoney	Login CancelTicket InternationalFlight DomesticFlight FlightInformation

Fig. 1: Proposed Approach Result

i.e. IR part, with the help of parsed keywords, we find the dependent CRC of class diagram with respect to change request. We used the extracted keywords of change request and retrieve impacted class names from the CRC of the system. Output of this program is names of the impacted classes with respect to the change request.

VI. EXPERIMENTAL RESULTS

In this we have shown the software change request and all the classes of that respective software. In the result we found the impacted class of the software which is affected by that requirement request. Our software change impact classes is shown in Table 1.

VII. FUTURE DIRECTION OF WORK

In the proposed approach we are using class diagram for impact analysis. To refine the SCIA, this approach may be applied in other UML models like in use case, sequence diagram, activity diagram. By analysis of these diagrams results will be fine grained. To find the impacted classes, we have used search based information retrieval approach. By using advance approaches like artificial intelligence etc., in information retrieval techniques, SCIA will be more efficient and improved.

ACKNOWLEDGEMENT

The authors are thankful to the Department of Computer Science & Engineering, United College of Engineering & Research, Allahabad, Up, India, for providing research facilities and their faculty for being the constant source of inspiration. The authors would like to thank co-author Associate Prof. Dharmendra kumar for his valuable support during the preparation of this paper.

VIII. CONCLUSION

Now we can say that clearly that, MDD plays an important for development and SCIA. The approach we have proposed reduces the effort that is unnecessary required in source code based SCIA. With the help of our approach we have developed, we can find the impacted classes in an early stage. SCIA is important activity of SCM process. Software system documentation and coding is increasing in massive rate, it is very difficult to impact analysis of whole system manually, for small systems manual SCIA might be useful,

but for Large System, SCIA is a tedious job. Our approach results have coarse granularity over the existing approaches. Benefits of our approach, at the class diagram level, save time and effort required in later stages of change management of the system. Thus our proposed approach allows efficient SCIA of the systems and reduces effort with a significant improvement over the traditional and older methods.

REFERENCES

[1] Antoniol, G. , Canfora, G. , Casazza, G. , Lucia, A. De and Merlo, E. (2002). Recovering traceability links between code and documentation. Software Engineering, IEEE Transactions on, 28(10):970983.

[2] Arnold, R. and Bohner, S.(1993). Impact analysis-towards a framework for comparison. In Software Mainte-nance CSM-93, Proceedings., Conference on, pages 292301.

[3] Breivold, H. , Crnkovic, I. and Larsson, M. (2012). A systematic review of software architecture evolution re-search. Information and Software Technology, 54(1):16 40.

[4] Briand, L. C., Labiche, Y. and OSullivan, L.(2003). Impact analysis and change management of uml models. In Proceedings of the International Conference on Software Maintenance, ICSM 03, pages 256,Washington, DC, USA,. IEEE Computer Society.

[5] Bengtsson, P. and Bosch, J.(1999). Architecture level prediction of software maintenance. In Software Mainte-nance and Reengineering, Proceedings of the Third Euro-pean Conference on, pages 139147. IEEE.

[6] Bohner, S. (1995). A graph traceability approach for software change impact analysis. PhD thesis, Fairfax, VA, USA,. UMI Order No. GAX95-42995.

[7] Choi, S., Park, S. and Sugumaran, V. (2012). A rule-based approach for estimating software development cost using function point and goal and scenario based require-ments. Expert Systems with Applications, 39(1):406 418.

[8] Corley, C. S., Kraft, N. A., Etzkorn, L. H. and Lukins, S. K. (2011). Recovering traceability links between source code and fixed bugs via patch analysis. In Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering, pages 3137. ACM.

[9] Fowler, M. (2003). UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3 edition.

[10] Goknil, A., Kurtev, I. and van den Berg, K. (2008). Change impact analysis based on formalization of trace relations for requirements.

[11] Khalil, A. and Dingel, J. (2013). Supporting the evo-lution of uml models in model driven software development: A survey.

[12] Kagdi, H. and Poshyvanyk, D. (2009). Who can help me with this change request? In Program Comprehension, 2009. ICPC09. IEEE 17th International Conference on, pages 273277. IEEE.

[13] Lu, H., Zhou, Y., Xu, B., Leung, H. and Chen, L. (2012). The ability of object-oriented metrics to predict change-proneness: a meta-analysis. Empirical Softw. Engg., 17(3):200242.

[14] Lehnert, S. (2011). 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, IWPSE-EVOL 11, pages 4150, New York, NY, USA. ACM.

[15] Li, Y., Li, J., Yang, Y. and Li, M. (2008). Requirement-centric traceability for change impact analysis: A case study. In Q.Wang, D. Pfahl, and D. Raffo, editors, Making Globally Distributed Software Development a Success Story, volume 5007 of Lecture Notes in Computer Science, pages 100111. Springer Berlin Heidelberg.

[16] Marcus, A., Sergeyev, A., Rajlich, V. and Maletic, J. I. (2004). An information retrieval approach to concept location in source code. In Reverse Engineering, 2004. Proceedings. 11th Working Conference on, pages 214223. IEEE.

[17] Pressman, R. (2001). Software Engineering: A Prac-titioners Approach. McGraw-Hill Higher Education, 5th edition.

[18] Rose, R., (2013). http://www-03.ibm.com/software/products/us/en/ratirosefami/. [Online; accessed 03-June-2013].

[19] Strens, M. and Sugden, R. C. (1996). Change anal-ysis: a step towards meeting the challenge of changing requirements. In Engineering of Computer-Based Systems. Proceedings., IEEE Symposium and Workshop on, pages 278283.

[20] Tao, Y., Dang, Y., Xie, T., Zhang, D. and Kim, S.(2012). How do software engineers understand code changes?: an exploratory study in industry. In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, FSE 12, pages 51:151:11, New York, NY, USA,. ACM.

[21] Tip, F. (1995). A survey of program slicing tech-niques. JOURNAL OF PROGRAMMING LANGUAGES, 3:121189.

[22] Visual paradigm. http://www.visual-paradigm.com/. [Online; accessed 03-June 2013].

[23] Wikipedia. PlagiarismWikipedia, the free encyclope-dia, 2013. [Online; accessed 15-June-2013].

