IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 4, April 2015.
www.ijiset.com
ISSN 2348 – 7968

Different Approaches using Change Impact Analysis of UML Based Design for Software Development

Ali Tariq Bhatti[1], Muhammad Murad Haider[2], Zill-e-Subhan[2]

[1]North Carolina A&T State University, Greensboro NC USA
atbhatti@aggies.ncat.edu, ali_tariq302@hotmail.com

[2]National University of Science and Technology (NUST), Islamabad Pakistan
murad.haider2003@gmail.com

[2]National University of Computer and Emerging Sciences (FAST), Lahore Pakistan
zsubhan@hotmail.com

Abstract

Because of rapidly changing technologies, requirements for the software systems are constantly changing. This requires a change in software design as well, as design should be traceable to the requirements. There is a strong need to deal with these changes in a systematic manner. For this purpose, proper decision making and change planning is required to effectively implement the change. Change Impact Analysis provides its services in this regard, by allowing us to assess the potential side - effects of change and also helps us in identifying that what is needed to be modified to accomplish the change. A number of impact analysis techniques have been proposed that perform impact analysis of UML based software design using a certain strategy and methodology. In order to explore the strengths and weaknesses of different approaches toward impact analysis, this survey paper includes an evaluation criterion for the comparison of different impact analysis techniques and a thorough analysis of these techniques based on evaluation criteria.

1. Introduction.

As software systems become increasingly large and complex, the need increases to predict and control the effects of software changes. Software can be modified to change the functionality of the software and services and this change also has wide effect on software design.The perception that software can easily be changed is often not the case, and understanding the complete impact of a modification on the software design is very important to managing the propagation of changes. The Evolving nature of software systems causes changes to the UML models involved in the design of a software project. So the impact of change must be analyzed for ensuring the consistency, to make these models up-to-date, and for assessing the impact of change on overall system [4].

Management of change in software involves, recognizing the need of change, evaluating the impact of proposed change to the whole system, adopting a technique for the accomplishment of change, performing initial and consequent modifications that are needed, and finally, managing the versions to collect change related data [7].

In case of a software design based on UML models, the changes can in turn lead to subsequent changes to other elements in the UML diagrams. Impact analysis is defined as the process of identifying the potential consequences (side-effects) of a change, and estimating what needs to be modified to accomplish that change [3, 4].

Design changes have a strong impact on the progress of a software development project and can have adverse effects, if these changes are not handled or analyzed in a systematic way. An issue in this regard is that, there is a lack of qualitative data, available to track the impact of such design changes. These changes in many cases increase the total development effort and delay the project completion [6].

ISSN 2348 – 7968

As changes in a software are introduced, avoiding the defects becomes increasingly labor intensive and error-prone. It exposes the systems and organizations, the software serves, to risks that are not visible without some impact analysis technique to effectively estimate and accomplish the change [1]. Whenever a change occurs in the software design, many other inconsistencies may be created in other design models, which are directly or indirectly affected by the change. Such propagation continues until change impact analysis identifies all inconsistencies [2]. Inconsistencies may be automatically modeled and detected by a set of consistency rules, also called well-formed ness rules in OMG (2001) [4].

In addition to consistency, dependency relationship between UML models is also an important factor to consider, achieving increased flexibility and expressiveness. Dependence relationship checking is also vital to achieve correct impact analysis results because checking of dependence relationship will help us to identify the elements that are directly or indirectly affected by the change.

Whenever a change occurs in UML based software design, impact analysis must be performed before implementation in order to assess the potential side effects of the change thus allowing early decision making and change planning [4]. Impact analysis of the change is necessary as it helps in estimating the cost of the change, to understand the relationship between the model, directly affected by the change, and the structure of the software (dependence relationship), and to determine the parts of the software that are needed to be tested after the implementation of change [5].

At design level, documentation of system's architecture is important as it aids in analyzing the impact of change. The main advantage of this documentation is its possible benefit as a tool when making changes to the system in the perspective of architectural evolution. The rationale for the architecture is fundamental information when analyzing how new or changing requirements impact the design [9].

In [10], a three – part frame work is proposed, which compares different impact analysis approaches. These parts correspond to the techniques to accomplish impact analysis and effectiveness of each technique.

In order to evaluate different impact analysis techniques, we have surveyed and analyzed various approaches, focusing on the impact analysis of changes in software design. The strengths and weaknesses of different techniques are assessed based on evaluation criteria, presented in the paper.

The rest of the paper is structured as follows: the succeeding section 2 contains discussion about different impact analysis approaches, and an evaluation criterion for comparison of different impact analysis techniques. Section 3 contains a detailed analysis of these approaches and section 4 outlines the main conclusions.

2. Impact Analysis Techniques.

Change impact analysis is about identifying the possible side effects of a change and estimating what needs to be modified to accomplish a change. To perform the impact analysis different authors have presented different approaches in order to assess the impact of change on the software development phases or artifacts.

The techniques that have yet been proposed to support change impact analysis, mostly support impact analysis at the code level of software systems, but little effort has been made for change impact analysis at the architectural or design level.

Different approaches to impact analysis are reported in literature that performs impact analysis of change in the software at using certain methodology. These techniques may vary from each other, on how these techniques use to accomplish impact analysis i.e. one technique may rely only on checking the consistency of UML diagrams , while another technique may focus on checking the dependence relationship among UML models along with consistency checking. Following is the overview of some approaches proposed by different authors:

2.1 Impact Analysis and Change Management of UML Models (L.C. Briand *et al* ,2003)

L. C. Briand *et al* [4] have proposed a UML based approach to impact analysis. It involves consistency checking of UML diagrams, detection of change according to some change taxonomy, determining with the help of formal rules (defined by OCL or UML meta model), the elements on which the changes have direct or

ISSN 2348 – 7968

indirect impact; after these steps a distance measure is computed to identify elements that are more likely to require a change, for prioritization of change impacts. This approach also involves a prototype tool to help automate the process of impact analysis.

2.2 Tracking the Impact of Design Changes during Software Development (Frank Padberg et al, 2001)

Padberg *et al* [6] have presented a methodology to gather data about design changes in an efficient way. It is proposed to use a "data base of design changes", consistently documenting changes, as these occur in a software development project. It will provide answers for various critical issues like assessing the impact of change and efforts for adjustments, this methodology works for any kind of software development project.

2.3 Automated impact analysis of UML Models (L.C. Briand *et al* ,2005)

L.C. Briand *et al* [3] have presented a methodology to perform impact analysis of UML based software design, which is identical to the one presented by L. C. Briand et al [4]. It involves consistency checking of models, change detection, determination of direct and indirect impact of change on model elements and calculation of distance measures. In addition two case studies about Automated Teller Machine (ATM) system and Cruise Control (CC) system are also presented, in which impact analysis is performed using prototype tool and at the same time as, accuracy of proposed approach is examined in another way (with respect to source code).

2.4 Supporting Impact Analysis and change Propagation in Software Engineering Environments (Jun Han, 1997)

Jun Han [7] has presented an approach to address the issues, concerning the change propagation and impact analysis in the context of a generic software engineering environment. The approach emphasizes on three aspects of change management that are; system representation, impact analysis, and change propagation. The main focus in this approach is on applying change propagation and impact analysis using original representations of software artifacts and dependencies instead of extracted system representations, using environment facilities like specification of consistency property, artifact manipulation, and defining relationships (dependencies).

2.5 Software Change Impacts An Evolving Perspective (Shawn A. Bohner, 2002)

Shawn A. bohner [1] presents an approach that focus an on going research on impact analysis of change in the systems that are increasingly using middle-ware and integrated Commercial- Off-The- Shelf (COTS) components. The focus in this research is on extending the current software change impact analysis technology (to support middle-ware and COTS) in order to incorporate interoperability dependency relationships and to provide effective method for navigating software changes using three dimensional (3D) visualization techniques, to address the emerging middle-ware problem.

2.6 Software Maintenance: An Approach to Impact Analysis of Object Change (Samuel Ajila, 1995)

Samuel Ajila [5] has proposed a technique in which a tool is presented to assess the impact of propagation of change (impact analysis). The model presented is independent of any language or design method and captures four software life cycle phases which are requirement, specification, design and coding. This technique focuses on two kinds of dependencies i.e. inter – phase dependency (dependency relation between objects of different phases) and intra – phase dependency (dependency relation between the objects of the same phase).

2.7 An Object - Based, Attribute - Oriented Approach for Software Change Impact Analysis (Chung-Yang Chen *et al,* 2007)

Chung-Yang Chen et al [2] have proposed a technique to accomplish change impact analysis in software project management. The technique employs an object – based, attribute oriented technologies to determine the artifacts that are affected by the change, and their relationships. It also involves the implementation of a prototype to support the automation of technique.

In order to compare different impact analysis techniques parameters that are considered crucial while comparing different techniques are Scope, Nature, Prototype/ Tool support, Case study, Consistency checking and Dependency Checking In the following, an evaluation criterion is presented for the analysis of aforementioned impact analysis techniques on the bases of parameters mentioned above:

ISSN 2348 – 7968

Evaluation Parameter	Description	Possible Values
Scope	Extent to which impact analysis is performed.	Name of the artifact or software engineering life cycle phase
methodology	Whether the technique is code based or model based	Code based, model based , both or N/A (if it is not defined)
Prototype /Tool support	Have the tool or prototype been developed to support automation?	Yes/ No
Case study	Whether the approach have been applied to an example or scenario.	Yes/No
Consistency Checking	Whether the consistency of artifacts is checked during CIA	Yes/ No
Dependency Checking	Whether Dependency relationship of artifacts is checked during CIA.	Yes/ No

Table 1: Evaluation criteria for change impact analysis techniques.

3. Analysis.

Following is the detailed analysis of the approaches presented by different authors, to perform Software Change Impact Analysis (CIA), based on critical evaluation parameters presented in Table 1.

Table 2 present the detailed analysis of different approaches related to software change impact analysis according to the evaluation parameters presented in table 1. Analysis shows that all the approaches involve consistency checking except the approaches presented by Frank Padberg et al [6], Shawn. A. Bohner [1], Chung- Yang Chen et al [2] and Samuel Ajila [5]. It is a limitation of these approaches because consistency is crucial in achieving correct results from impact analysis algorithm.

An important issue here is to consider the methodology employed by impact analysis techniques i.e. whether a technique applies model based or code based impact analysis method or both. To perform impact analysis using code based method, the change should be implemented. Whereas model based methods allow the accomplishment of impact analysis before the implementation of change. So model based methods are advantageous in the sense that they ensure earlier decision making and change planning. However, code based methods yield more accurate results because the implementation of the components is known.

From the detailed analysis performed in table 2, it can also be found that all the approaches, except Frank Padberg et al [6], Shawn. A. Bohner [1], and Samuel Ajila [5], offers the implementation of the approach on some case study or real world examples which ensures the applicability and practicality of these approaches. So, there is a room for improvement to ensure the practicality and to enhance the understandability of their approaches, above mentioned authors should also include the application of their approaches on some case studies or real world examples to validate both the implementation and methodology.

All the approaches, except the approach presented by Frank Padberg et al [6], perform dependency relationship checking. It is also vital to achieve correct impact analysis results. Because checking of dependence relationship will help us to identify the elements that are directly or indirectly affected by the change, as an element can have indirect impact of change if it is dependent upon the element which is directly affected by the change or in which the change occurs.

ISSN 2348 – 7968

All impact analysis techniques that are analyzed, are automatable which provides room for automated tool support for these approaches, in order to automate the impact analysis process. And also tools or prototypes for some of the approaches have already been developed.

Serial #	Technique	Scope	Methodology	Prototype/Tool Support	Case study	Consistency checking	Dependency Checking
1	L.C. Briand *et al* (2003)	UML based design models	Model based	Yes	Yes	Yes	Yes
2	L.C. Briand *et al* (2005)	UML based design models	Model based	Yes	Yes	Yes	Yes
3	Frank Padberg *et al*	Software design	N/A	No	No	No	No
4	Jun Han(1996)	Software development/main tenance	Code based	No	Yes	Yes	Yes
5	Shawn A.Bohner(2000)	CBSE COTs and middle-ware	Code based	No	No	No	Yes
6	Chung-Yang Chen et al (2007)	Software development life cycle	Both	Yes	Yes	No	Yes
7	Samuel Ajila (1995)	Software requirement, specification, design, programming	Code based	Yes	No	No	Yes

Table 2: Analysis of change impact analysis approaches

ISSN 2348 – 7968

4. Conclusion

Design changes occur frequently during software development so the impact analysis of change in software design has gained considerable attention. With the growing use of UML models for software design, the design of software systems involves a large number of interdependent UML diagrams. Because of inevitability of change in software systems, these diagrams also suffer from changes, to deal with changes in the requirements, and which in turn leads to subsequent changes in other dependent elements in the UML diagrams. The introduction of impact analysis of changes in software design then adds more fidelity to change visibility. It enables more accurate identification of side-effects of change and identifies that what is needed to be done in order to implement the change. Various impact analysis techniques have been proposed that perform impact analysis of change in UML based software design using a certain strategy and methodology (code based or model based). In this survey paper, we have performed a thorough survey of different impact analysis techniques, on the basis of an evaluation criterion which defines certain parameter that are crucial to compare these different techniques. Results of the analysis reveal that consistency checking and dependency checking is not being performed in certain impact analysis techniques but this feature is very crucial to get accurate impact analysis result. In addition, all impact analysis techniques that are analyzed are automatable that provide room for the development of automated tool to support the automation of these techniques. Furthermore these impact analysis approaches have same purpose but methodology employed by these techniques, to perform impact analysis, is different i.e. some approaches perform impact analysis using code based methods while other accomplish impact analysis using model based methods.

References

[1] Bohner, S.A., "Software change impacts -an evolving perspective", *Software Maintenance, 2002. Proceedings. International Conference on* , vol., no., pp. 263-272, 2002

[2] Chung-Yang Chen; Cheung-Wo She; Jia-Da Tang, "An object-based, attribute-oriented approach for software change impact analysis", *Industrial Engineering and Engineering Management, 2007 IEEE International Conference on* , vol., no., pp.577-581, 2-4 Dec. 2007.

[3] Briand, L. C., Labiche, Y., O'Sullivan, L., and Sówka, M. M. 2006. "Automated impact analysis of UML models". *J. Syst. Softw.* 79, 3 (Mar. 2006), 339-352. DOI= http://dx.doi.org/10.1016/j.jss.2005.05.001

[4] Briand, L. C., Labiche, Y., and O'Sullivan, L. 2003. "Impact Analysis and Change Management of UML Models". In *Proceedings of the international Conference on Software Maintenance* (September 22 - 26, 2003). ICSM. IEEE Computer Society, Washington, DC, 256.

[5] Ajila, S. 1995. "Software maintenance: an approach to impact analysis of objects change", *Softw. Pract. Exper.* 25, 10 (Oct. 1995), 1155-1181.DOI=http://dx.doi.org/10.1002/spe.4380251006

[6] F. Padberg, "Tracking the impact of design changes during software development", *EDSER-3*, May 2001, pp. 50-55.

[7] J. Han, "Supporting Impact Analysis and Change Propagation in Software Engineering Environments", *Proc. STEP'97 - 8th International Workshop on Software Technology and Engineering Practice*, London (July 1997), pp. 172-182.

[8] L. Briand, Y. Labiche and G. Soccar, "Automating Impact Analysis and Regression Test Selection Based on UML Designs" ,Carleton University, Technical Report SCE-02- 04, http://www.sce.carleton.ca/Squall/Articles/TR_SCE-02-04.pdf, March, 2002, a short version appeared *in the proceedings of ICSM* 2002

[9] Bratthall, L., Johansson, E., and Regnell, B. 2000. "Is a Design Rationale Vital when Predicting Change Impact?" A Controlled Experiment on Software Architecture Evolution. In *Proceedings of the Second international Conference on Product Focused Software Process Improvement* (June 20 - 22, 2000). F. Bomarius and M. Oivo, Eds. Lecture Notes In Computer Science, vol. 1840. Springer-Verlag, London, 126-139.

[10] Arnold, R. S. and Bohner, S. A. 1993. "Impact Analysis - Towards a Framework for Comparison". In *Proceedings of the Conference on Software Maintenance* D. N. Card, Ed. IEEE Computer Society, Washington, DC, 292-301.

ISSN 2348 – 7968

BIOGRAPHIES

Ali Tariq Bhatt received his Associate degree in Information System Security (Highest Honors) from Rockingham Community College, NC USA, B.Sc. in Software engineering (Honors) from UET Taxila, Pakistan, M.Sc in Electrical engineering (Honors) from North Carolina A&T State University, NC USA in 2010, and currently pursuing PhD in Electrical engineering from North Carolina A&T State University. His current research interests include Coding Algorithm, Networking Security, Mobile Telecommunication, Biosensors, Genetic Algorithm, Swarm Algorithm, Health Bioinformatics, Control system, Power, Software development, Communication, and Signal Processing.