International Journal of Computer Applications (0975 – 8887)
Volume 6– No.6, September 2010

Dependency based Process Model for Impact Analysis: A Requirement Engineering Perspective

Chetna Gupta
Jaypee Institute of Information Technology
A-10 Sector – 62
Noida, India

Yogesh Singh
University School of Information Technology
Guru Gobind Singh Indraprastha University, Kashmere Gate
Delhi, India

Durg Singh Chauhan
Uttarakhand Technical University
Dehradun, India

ABSTRACT

Changing requirements of customer needs establishes the need to analyze impact of requirement changes. For success of any software requirement analysis is very essential. In this paper, we propose a four stage method engineering process which aims at estimating impact of change. The process model described is a linear layered model. Impact sets are computed by analyzing dependency tractability relations with other connected method components. The results produced provide two type of information (a) added, deleted, modified methods (b) depth (extent) of impact on the system.

General Terms

Change impact analysis, software maintenance, software testing.

Keywords

Change impact analysis, method, method engineering, requirements traceability, situational method engineering, software testing.

1. INTRODUCTION

Software evolution is an ongoing process carried out in order to meet changing requirements of stakeholders such as beneficiaries or users. For the success of any software, requirements analysis is critical. In real-world projects requirements change throughout the project due to changing user requirements and application goals.

Requirement engineering (RE) according to [12] is "*a sub discipline of systems engineering and software engineering that is concerned with determining the goals, functions, and constraints of hardware and software systems*." Any change in requirement will accordingly affect design, coding and implementation. To cope up with the situation test cases are to be adapted in order to test implementation against revised requirements. Thus there is a need to analyze impact of requirement changes on other requirements leading to design, coding and implementation in order to understand likely impact of requirement changes on product quality and need for re-testing.

We propose a four stage method engineering process. The requirement engineering phase consists of representation of design phase as structural base and construction phase as dependency base which consists of organization (detail of co-relation among methods) of method structure and dependencies between them. In order to analyze impact of change we use concept of Situational Method Engineering (SME) [5] which assumes existence of a method repository from where method (s) of interest are retrieved, modified or assembled into a new method that is subsequently stored in repository.

The proposed approach computes impact set by analyzing change in requirements. Dependency relations are used as trace links to determine depth of impact on the system. The paper is structured as follows. Section 2 gives details of related work of our work. Section 3 presents the proposed framework and process model. In Section 4, we illustrate the whole process with the help of an example. In Section 5, we provide result analysis to illustrate change impact analysis. Section 6 presents the result analysis of experimental setup conducted on a set of programs to validate the presented approach. Section 7 concludes the paper.

2. RELATED WORK

The main component of our approach is estimation of impact of change in the requirement phase of software development. Many researchers have addressed the issue of change impact analysis in the context of requirements modeling. [6, 17] use UML profiling mechanism for goal-oriented requirements engineering approach whereas KAOS model of [19] can be represented in UML by using the approach given in [6]. [17] use UML profiling mechanism to provide an integrated modeling language for functional and non-functional requirements that are mostly specified by using different notations. [20] describes a tool support for graphical requirement models and automatic generation of Software Requirements Specifications (SRS) for checking constraint violations for requirements models. However they do not support change impact analysis upon requirements and their relations. [15] proposes a model for requirement traceability which captures relations between different software artifacts and requirements instead of relations between requirements. [16] presents an approach to capture more precise traces by defining operational semantics - with a triplet (event, condition, actions) for traceability in UML. [21] proposes a generic solution for both specification and appliance of traceability. [18] illustrate the need for developer tolerance of inconsistencies for managing inconsistencies between different model artifacts but it does not provide any techniques to determine the impacts within a model. [2] presents a tracing mechanism based on SysML UML 2.0 profile to define requirements according to a proposed requirements classification. [10] uses an XML based approach of traceability graph to detect the dependency between model elements. [9] uses graphs and sets to represent changes. [1] defines elements and relations between elements to be traced with intra-level and inter-level dependencies. [7] uses transitive closures of call graphs to estimate impact analysis. [8] discusses an impact analysis based on traceability data of an object-oriented system by tracing across phases with intra-level and inter-level dependencies. Whereas change impact analysis for software architectures is discusses in [22], although their analysis is restricted to architectural level only.

3. PROPOSED APPROACH

We propose a two layered framework to break situational method engineering task into stages. This is organized in a hierarchy of abstractions shown in Figure 1 below.

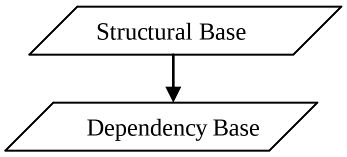

Figure 1. Two-layered framework

As stated in [3] a change introduced can be in one of two phases "*A proposed change implies that impact analysis should be performed to determine how change would impact the existing system, whereas an implemented change implies that all impacted artifacts and their related links should be updated to reflect the change*". In our approach we aim at estimating the impact of change (and its depth) about the possible impacts of a proposed requirements change on the overall system by analyzing methods components and their dependency relations.

3.1 Structural Base

This level provides the elaborative design of the process for which changes are to be made. In other words it reveals relationship between method components. It determine how the method is constructed, what features it provides, what constraints are applicable etc. This level provides components of selected method and hence component of new method can be identified by looking at structure (architecture) of selected method. If structure matches with components of desired new method then next level of abstraction is reached. If in case it fails to match, desired new method can be constructed by using either of two approaches (a) method engineer can add any of missing component and corresponding dependencies can be worked out (b) method engineer can delete any of unwanted component from selected architecture and accordingly can work out for dependencies between remaining components. To construct structural base for a given method we use the concept of [11] and change in two structures are identified using design given in Figure 2. It takes detailed design of method components and their relationship as input for both the processes (programs for which these method components are constructed). These are processed in input-output processing model which compares two method components to produce changed set. This changed set compromises of those methods which are deleted, added or modified. Software maintainer has to analyze this changed set to estimate impact of change. To study this impact further the next level of abstraction is approached which is a meta-models. Here dependencies between method components are worked out for analyzing relationship contained in each complex method block of structural base. The details are given in following section.

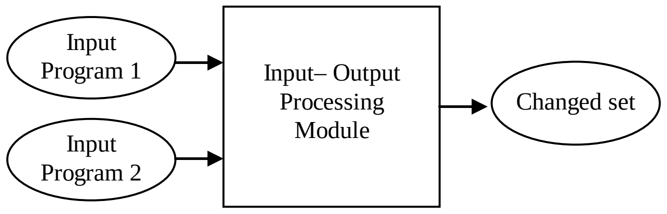

Figure 2. Detailed view of Structural Base

3.2 Dependency Base

Dependency base is the second level of abstraction. It is a broad structure of the method and relies on a model of a method. This may be a meta-model, for example the fragment [5], contextual [4], or decisional meta-model [13], or alternatively it can be based on a generic model [14]. To define dependency between method primitives of structural base we use generic model of methods approach given in [14].

3.3 Process Model

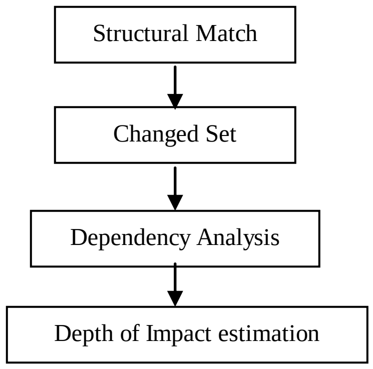

Figure 3. Process Model

We organize whole process in four stages as shown in Figure 3. The process starts with structural matching stage. It is assumed that components of old and new methods and their relationships have been worked out before processing for structural match. The matching takes place as follows: The two programs are taken as input and are compared with the aim of producing dissimilar components. The second stage considers set of methods selected (dissimilar methods) in first stage. Third stage aims to determine dependencies among methods between (a) selected methods of changed set at second stage, (b) any missing ones determined, and (c) any new ones added. The last stage uses information of third stage to calculate depth of impact. Dependency based traces are used to gather this information.

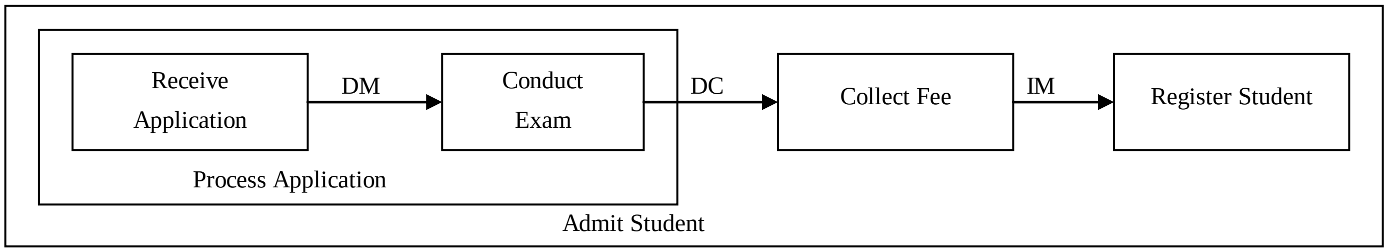

Figure 4. Components description and their relationship for Admit Student process

4. ILLUSTRATION OF PROCESS

To illustrate our process, let us assume the process of admitting a student to a university. As shown in Figure 4 above, Admit Student is a complex method built from three methods, Process Applicant, Collect Fee and Register Student respectively. The structure shows that Process Applicant method is complex and built over Receive Application and Conduct Exam. These two components are in Deferred-Must (DM) dependency. That is after the application has been received, Conduct Exam can be enacted any time later but it must be enacted. The method, Collect Fee, is dependent on enactment of Process Applicant. Once it is enacted, Collect Fee is enacted in a Deferred-Can (DC) mode. This is because Admit Student assumes that some students may not pay fee, possibly because they have got admission elsewhere (necessity is Can) and also that there can be a time gap between student selection and fee payment (Urgency is Deferred). Lastly, Register Student must be done immediately after Collect Fee. So, it is in an Immediate-Must (IM) dependency with Collect Fee.

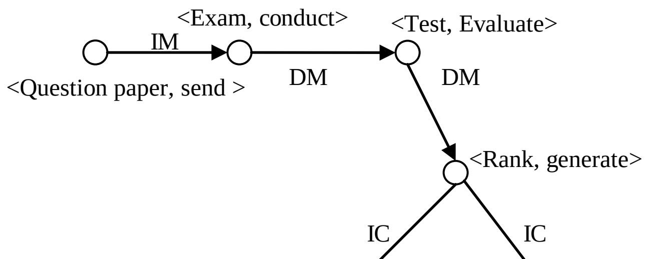

Figure 5. Detailed view of Conduct Exam method

Conduct Exam method in Figure 5 starts off by sending question papers, <question paper, send> method primitive. The conduction of exam must be immediately done after arrival of question paper. This is captured by node <Exam, conduct> and IM dependency. The evaluation of test can be done anytime after conduction of test. This is captured by node, <Test, Evaluate> and DM dependency. The rank can be generated anytime after evaluation of answer sheets is done, this is captured by node <Rank, Generate> and DM dependency. Now, based on rank of candidate, student can either be called for counseling or he can be disqualified. This is captured by node <Student, call> and <Student, Disqualify> respectively with (Immediate Can) IC dependency in both the cases. Dependency relationship among method primitives is shown in Table 1 below.

Table 1. Dependency relationship among method primitives

S No.	Method Primitive I	Method Primitive II	Type of dependency
1	<question paper, send>	<Exam, conduct>	IM
2	<Exam, conduct>	<Test, Evaluate>	DM
3	<Test, Evaluate>	<Rank, Generate>	DM
4	<Rank, Generate>	<Student, call>	IC
5	<Rank, Generate>	<Student, Disqualify>	IC

Now consider that there is a requirement of changing process of conducting an exam. The stakeholder wants to add one more component of conducting an exam by online process to existing process. According to specified requirement a student can now appear for exam online and it will receive result immediately. This will be helpful for him/her to estimate whether he or she will be admitted to course or not. On the other hand a student who opts for offline exam has to wait for some period of time for their results. On looking to requirements software developers proposed a modified view of conduct exam similar to Figure 6 shown below. As a part of impact analysis it will be beneficial to estimate impact of change on overall system at this stage only so that after effects of making changes do not affect design, coding and implementation of the system.

Figure 6 shows modified model of method components and their dependencies. Here Conduct Exam method starts off by sending question papers, <question paper, send> method primitive for both online and offline exam conduction. In both the cases conduction of exam must be immediately done after arrival of question paper. This is captured by nodes <Offline Exam, conduct> and <Online Exam, conduct> with IM dependency. The evaluation of test can be done anytime after conduction of test in case of offline exam whereas it has to be done immediately in case of online exam. These two events are captured by node, <Test, Evaluate> and DM dependency for former approach and with IM dependency for latter. The rank can be generated anytime after evaluation of test is done in case of offline exam and this is captured by node <Rank, Generate> and DM dependency whereas it has to be done immediately in case of online exam and is captured by IM dependency. Now, based on the rank of candidate, student can either be called for counseling or he can be

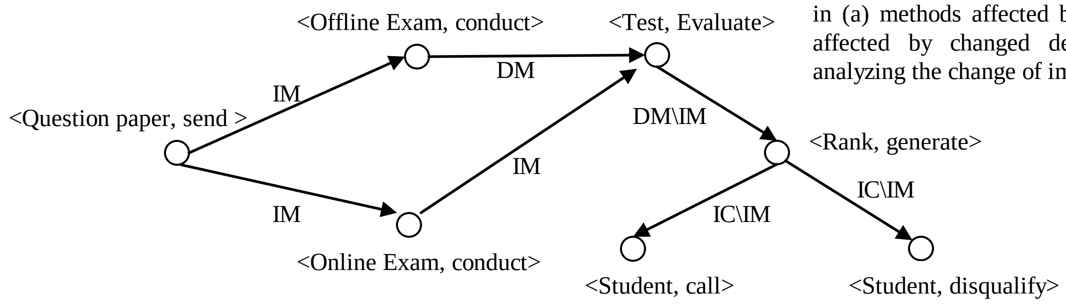

Figure 6. Modified view of Conduct Exam method

disqualified. This is captured by node <Student, call> and <Student, Disqualify> respectively with IC dependency in both the cases for online exam and with IM dependency for both the cases of online exam. Dependency relationship among method primitives of modified process are shown in Table 2 below.

Table 2. Dependency relationship among method primitives of modified process

S No.	Method Primitive I	Method Primitive II	Type of dependenc y
1	<question paper, send>	<Offline Exam, conduct>	IM
2	<question paper, send>	<online Exam, conduct>	IM
3	<Offline Exam, conduct>	<Test, Evaluate>	DM
4	<online Exam, conduct>	<Test, Evaluate>	IM
5	<Test, Evaluate>	<Rank, Generate>	DM/IM
6	<Rank, Generate>	<Student, call>	IC/IM
7	<Rank, Generate>	<Student, Disqualify>	ICIM

5. RESULT ANALYSIS

To estimate impact of change software maintainer's analyses dependencies worked out between method primitives of base process and modified process shown in Table 1 and Table 2 respectively. The results of comparison are shown in Table 3.

Impact set is generated on basis of change in dependency in existing method primitives and added method primitives. Table 4 represents impact set for added method primitives and changed dependency method primitives based on dependency traceability. The results of Table 4 indicate suspicious method primitives that need to be tested. The results also help in classifying information in (a) methods affected by added dependency and (b) methods affected by changed dependency. This will be helpful in analyzing the change of impact at early stage only.

5.1 Depth of Impact

It can be calculated from dependency base by analyzing number of methods affected (addition, deletion or change in dependency between methods). The level of depth will be last node where change has propagated. As shown in Figure 6 the extent of the impact is till end. Hence it can be concluded that depth of impact is till the last level.

6. CONCLUSION

In this paper, we proposed a change impact analysis technique based on dependency traceability of requirements relations. To achieve our aim we propose a framework for process model for comparing two program requirements. Our assumption is that if there is to be a rejection in making desired changes by stakeholders, then it is less expensive to do this early in engineering process rather than later. The impact sets are calculated by tracing dependency relation among method components.

The engineering process described here is a stage-wise linear model. Once structural matching is done, change method components for changed set are highlighted. To analyze the method components more deeply the dependency relations are analyzed in dependency base. This will help in gathering impact sets based on dependency traceability relations among methods. Based on this data the depth (extent) of impact on the system can be calculated.

Table 3. Comparison of Results

Method Primitive I of Process 1	**Method Primitive II of Process 1**	**Type of dependency**	**Method Primitive I of Process 2**	**Method Primitive II of process 2**	**Type of dependency**	**Change in dependency**
- -	--	--	<question paper, send>	<online Exam, conduct>	IM	Added
--	--	--	<online Exam, conduct>	<Test, Evaluate>	IM	Added
<Test, Evaluate>	<Rank, Generate>	DM	<Test, Evaluate>	<Rank, Generate>	DM/IM	Yes
<Rank, Generate>	<Student, call>	IC	<Rank, Generate>	<Student, call>	IC/IM	Yes
<Rank, Generate>	<Student, Disqualify>	IC	<Rank, Generate>	<Student, Disqualify>	ICIM	Yes

Method Primitive I of Process 1	**Method Primitive II of Process 1**	**Type of dependency**	**Method Primitive I of Process 2**	**Method Primitive II of process 2**	**Type of dependency**	**Change in dependency**
<question paper, send>	<Exam, conduct>	IM	<question paper, send>	<Offline Exam, conduct>	IM	No
- -	--	--	<question paper, send>	<online Exam, conduct>	IM	Added
<Exam, conduct>	<Test, Evaluate>	DM	<Offline Exam, conduct>	<Test, Evaluate>	DM	No
--	--	--	<online Exam, conduct>	<Test, Evaluate>	IM	Added
<Test, Evaluate>	<Rank, Generate>	DM	<Test, Evaluate>	<Rank, Generate>	DM/IM	Yes
<Rank, Generate>	<Student, call>	IC	<Rank, Generate>	<Student, call>	IC/IM	Yes
<Rank, Generate>	<Student, Disqualify>	IC	<Rank, Generate>	<Student, Disqualify>	ICIM	Yes

Table 4. Suspicious method primitives

7. REFERENCES

[1] Ajila, S., 1995. Software maintenance: An approach to impact analysis of object change. Software - Practice and Experience.

[2] Albinet, A., Boulanger, J.L., Dubois, H., Peraldi-Frati, M.A., Sorel, Y., Van, Q.D., 2007. Model-Based Methodology for Requirements Traceability in Embedded Systems. ECMDA TW 2007 Proceedings, Haifa.

[3] Cleland-Huang, J., Chang C.K., Christensen M., 2003. Event-Based Traceability for Managing Evolutionary Change. IEEE Transactions on Software Engineering.

[4] G. Grosz, C. Rolland, S. Schwer, C. Souveyet, V. Plihon, S. Si-Said, C. Ben Achour and C. Gnaho, 1997. Modelling and Engineering the Requirements Engineering Process: An Overview of the NATURE Approach, Requirement engineering Journal.

[5] Harmsen F., Brinkkemper S., Han J.L.O., 1994. Situational Method Engineering for Information System Project Approaches,. Proceedings of the IFIP WG8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle. Elsevier Science Inc.

[6] Heaven, W., Finkelstein, A., 2004. UML Profile to Support Requirements Engineering with KAOS. IEE Proceedings – Software.

[7] Law, J., Rothermel, G., 2003. Whole program path-based dynamic impact analysis. International Conference on Software Engineering (ICSE`03).

[8] Lindvall, M., Sandahl, K., 1998. Traceability aspects of impact analysis in object-oriented systems. Software Maintenance: Research and Practice.

[9] Luqi: A, 1990. Graph Model for Software Evolution. IEEE Transactions on Software Engineering.

[10] Maletec, J.I., Collard, M.L., Simoes, B., 2005. An XML based Approach to Support the Evolution of Model-to-Model Traceability Links. Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering.

[11] Naveen Prakash, Maneeesha Srivastav, Chetna Gupta, Vipin Arora, 2007. An intention driven method engineering approach", First International Conference on Research Challenges in Information Science (RCIS).

[12] Phillip A. Laplante, 2007. What Every Engineer Should Know about Software Engineering.

[13] Prakash N., 1999. Towards a Formal Definition of Methods, Requirements Engineering Journal, Springer.

[14] Prakash Naveen, 2006. On Generic Method Models, Requirements Engineering Journal.

[15] Ramesh, B., Jarke, M., 2007. Toward Reference Models for Requirements Traceability. IEEE Transactions on Software Engineering.

[16] Reshef, N. A., Paige, R. F., Rubin J., Shaham-Gafni Y., Kolovos D.S., 2005. Operational Semantics for Traceability. ECMDA TW 2005 Proceedings.

[17] Supakkul, S., Chung, L., 2005. A UML Profile for Goal-Oriented and Use Case-Driven Representation of NFRs and FRs. SERA'05.

[18] Van Gorph, P., Altheide, F., Janssens, D., 2006. Traceability and Fine-Grained Constraints in Interactive Inconsistency Management. ECMDA TW 2006 Proceedings, Bilbao.

[19] Van Lamswerdee, A., 2001. Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice. Invited Minitutorial, Proceedings RE'01 - 5th International Symposium Requirements Engineering.

[20] Vicente-Chicote, C., Moros, B., Toval, A., 2007. REMM-Studio: an Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting. In Journal of Object Technology, Special Issue TOOLS Europe.

[21] Walderhaug, S., Johansen, U., Stav, E., 2006. Towards a Generic Solution for Traceability in MDD. ECMDA TW 2006 Proceedings, Bilbao.

[22] Zhao, J., Yang, H., Xiang, L., Xu, B., 2002. Change impact analysis to support architectural evolution. Journal of Software Maintenance and Evolution: Research and Practice.