Dependency based Technique for Identifying the Ripple Effect of Requirements Evolution

YuQing Yan
Sun Yatsen University/Department of Computer Science,Guangzhou,China
Guangdong University of Foreign Studies/Cisco School of Informatics,Guangzhou,China
Email: yyqelsa@yahoo.com.cn

ShiXian Li
Sun Yatsen University/Department of Computer Science,Guangzhou,China
Email: lnslsx@mail.sysu.edu.cn

WeiJun Sun
Sun Yatsen University/Department of Computer Science,Guangzhou,China
Guangdong University of Technology/Faculty of Computer,Guangzhou,China
Email: gdutswj@gdut.edu.cn

***Abstract*—Requirements evolve continuously and inevitably. In order to effectively manage requirements change, an understanding from a quantitative perspective is needed in determining the extent of the propagation of the requirements as they evolve. In this paper, we first look back at the impact analysis in the software life cycle and give an overview of requirements dependencies. We present a generic algorithm for identifying requirements dependencies and the definitions that are adopted for this study.Using matrix theory on the requirements dependencies, we present the formulas of the ripple effects of requirements evolution and their properties. A typical example in structured analysis of M.Jackson is described to show the effectiveness of the method presented here.**

***Index Terms*—requirements evolution; dependency; ripple effects**

I. INTRODUCTION

Requirements evolve continuously and inevitably[1], making evolution an inherent feature of requirements and software. The evolution of requirements may cause increased budgets, delayed schedules, as well as degraded systems. Hence, it is considered one of the most important issues in developing computer-based systems[2-6]. When a requirement changes, the "ripple effects" associated with the change will probably occur[3-7]. Due to existing dependencies between requirements, requirements are not isolated. When they change, they may create new requirements, and introduce conflicts with other existing requirements. In this case, all related requirements must be identified and modified so that the new functionality can coexist with the other requirements. Before formal change requests are implemented, all related requirements should first be found, otherwise there would be increased risks of over budget, delayed delivery and low quality software. If all requirements change requests have to be implemented, then the costs and schedules should be taken into account. These activities belong to change management, which is one of the most important aspects for a successful software development project. Therefore, some effective mechanisms[7] should be developed to manage the process of requirement change.

In the next sections of this paper, we look back at the impact analysis in the software life cycle, and give an overview of requirements dependencies. An generic algorithm for identifying requirements dependencies is presented, as are the definitions that are adopted for this study. Based on this algorithm and using matrix theory, the formulas of ripple effects of requirements evolution are then presented, and some quantitative properties about adjacent matrix and reachability matrix are described, and an example that is often used in structured analysis of M.Jackson is described. Finally we present our conclusions and further research issues on ripple effects of requirements evolution.

II. RELATED WORKS

Impact analysis research first came from software maintenance, which can be traced to a paper by Haney in 1972[8].The paper described a method that modeled the stability of large systems as a function of its internal structure and used a matrix to record the probability that a change in one module would trigger changes in other

Manuscript received July 27, 2011; revised September 18, 2011; accepted September 27, 2011.
Corresponding author: Yuqing Yan

© 2012 ACADEMY PUBLISHER
doi:10.4304/jsw.7.3.544-550

modules in the system. With software becoming more and more important for people and society , its applications have deepened and broadened into all aspects of the world. The scale and complexity of software have grown, which has resulted in difficulties of maintaining software. Changing requirements of organizations and users, low quality software, and changing environments of software and hardware have directly resulted in frequent modifications of software. Therefore, impact analysis has become a major research in software maintenance. Queille,Voidrot,Turver and Munro suggested[9] that impact analysis should be done as early as possible in the requirements level, not just on the code. Since 1994, impact analysis of requirements change and evolution has been proposed, while requirements engineering has become an important discipline in the area of software engineering.

Impact analysis of requirements change is used to identify the potential consequences of requirements changes and to estimate what needs to be modified to accomplish a change[10]. This is an essential part of change management. Without adequate analysis, it is not possible to confidently determine the extent, complexity and cost of proposed changes to a software system. The nature of requirement volatility will cause many requirement problems, such as inconsistency, defects, risk and ripple effects, and other side effects. There are four main topics in the research of impact analysis of requirements change in the literature. First is the management of inconsistencies caused by requirements changes. Bashar Nuseibeh, Steve Easterbrook and Hunter[11-16] have done much research in inconsistency management. The steps of the inconsistency management process are first detecting inconsistency(by monitoring for inconsistency), then diagnosing and handling detected inconsistency, recording the consequences of the inconsistency handling actions, and finally, continuously monitoring and evaluating inconsistency as development conditions change. Second is the detection and modification of the defects and errors induced by requirement changes. Yan[2] analyzed and classified the causes of requirement defects from two aspects both social and technical. They proposed a defect management process, and pointed out that quantitative analysis of requirement defect lists could be beneficial to evaluate the performance of engineers, define the degree of maturity of the requirement process, and analyze the impact of requirement changes. Based on a case study to collect industry data, Javed, Maqsood and Durrani [17], analyzed requirement changes in the design of systems, and at the later stages of software development. They found that requirement changes had a significant impact on the high severity defects of software. They classified requirement changes and defects, and analyzed the data by statistical method. Third is the analysis of the risk triggered by requirement changes. Strens and Suqden [10] suggested that change analysis should consider the requirement risks and there is a need to collect information to enable sensitivity and impact analysis to provide an effective means to avoid bad effects caused by changes. Byron J. Williams [7] presented some ways in the impact analysis of requirement changes to understand the risk management. The final is the identification of ripple effects of requirement changes. Based on matrix calculations, Yan [3] presented a method to define and calculate the requirements evolution ripple-effects, to confirm, as much as possible, the extent and scope of propagation of requirement changes, and to not allow any possible requirements defects as the sources of risk.

In impact analysis techniques, one kind of method is based on dependence analysis [18-20]. These researchers attempted to assess the resulting changes on semantic dependencies among program entities. Hassine, Rilling and Hewittbased, based on dependency analysis at the level of requirements, presented a technique to analyze change impacts of scenario and component[20]. Åsa G. Dahlstedt and Anne Persson[21] said that most of the interdependency types discussed are related to requirements management and requirements evolution.

III. An Overview Of Requirements Dependencies

"As systems grow ever bigger and more complex, are developed globally and become more interconnected, the impacts of changes are harder to analyze and control"[1]. So developing effective mechanisms to analyze the impacts of requirements changes is needed. One of the mechanisms is to address studying requirements dependencies and its application.

Requirements depend on and affect each other, and are not isolated from each other[21]. The complex relationships among requirements are a main cause for ripple effects with requirements evolving. About requirements dependencies (interdependencies), in 2003 Åsa G. Dahlstedt and Anne Persson[21] gave a detailed overview by outlining the current state of literature from an early dependency model of Pole's[22](related to requirements traceability), Carlshamr-e's model[23](used to plan release) and some other relating literature to their preliminary results. They pointed out some problems in the research of requirements dependency, the reason why little was known about the nature of requirements interdependencies, in what activities that requirements dependencies could engage. And they addressed the function and mechanism of dependencies in the impact analysis of requirements changes and management.

There are four main components for the study of requirements dependencies[21]: First, classifying requirements dependencies. The different types of dependencies that exist between requirements should be concerned and classified classified. Second, identifying requirements dependencies after classification of dependencies. This is difficult, but important. Third, recording a number of different dependencies after their indentifications.Fourth, using requirements dependencies in different activities such as requirements

© 2012 ACADEMY PUBLISHER

management, change management, reuse of requirements, release planning, testing, maintenance. The classification of requirements dependencies is the most important, and is the base of subsequent studies about requirements dependencies. Current literature tells us that there are several different schools in classifying dependencies.

The classification of requirements dependencies is important above all, it is the base of subsequent studies about requirements dependencies. Ref.[21] and other relatinting literature indicated that there were several different schools in classifying dependencies and the reasons why these schools could not be unified. We have synthesized these results into a new classification model from five aspects[24]: structural, implementation, lexeme, economics and evolution. In this model, ten different kinds of requirements dependencies are defined. They are Hierarchy, Horizon, Attribute, Similarity, Reference, Constraint, Cost/Value, Exclusion, Risk, Evolution. Some progress has been made in this model, such as: 1) Extending current models and adding some new types of dependencies. For example, Constraints can show how functional requirements depend on non-functional requirements, Conditional Reference shows how business rules work; 2) Enhancing the identification ability of the model. It can be used to identify some requirement dependencies that other models could not identify; 3) Strengthening the ability of expression of the model. It can be used to build dependencies between requirements of any level and any granularity; and 4) Simplifying some classifications of current models.

This new model is more robust and rational, and easily used to identify dependencies, as was shown previously[24].

IV. The Formulas to Calculate The Ripple effects of Requirements Changes

We define ripple effects of requirements changes as the effects caused by making some requirements changes which affect many other requirements or software components[25].

A. An Generic Algorithm of Identifying Requirements Dependencies

After the classification of dependencies is confirmed, it is important to build identification algorithms to identify all kinds of dependencies. This can be difficult for some kinds of dependencies, such as Reference and Similarity dependencies which should have developed some mechanisms to support developing such algorithms[24]. Reference dependency is decided by business rules, which depend on specific projects and domains. Similarity dependency needs a support of retrieval technology of natural language. Now, we are not yet ready to use specific technologies to develop identification algorithms for each dependencies, but we will provide a generic algorithm for a dependency: d-Associated Algorithm.

Algorithm 4.1 d-Associated Algorithm:

Begin

(1) Given d is a dependency, $\mathcal{R}$ is a set of requirements, $R \in \mathcal{R}$ is a requirement, then
(2) According to the definition of d, find a set of all related requirements: $R|d=\{R_1,R_2,\ldots,R_n\}$.
(3) If $R|d=\Phi$, then go to (8);
(4) Do Repeat
(5) If for each i, $i \geq 1$, $R_i \mid d=\Phi$, then go to (8).
(6) For each i, $i= \geq 1$,find all

```
{
Ri|d={R_i^{i1},R_i^{i2},...,R_i^{im}};
  For j=1 to m,
    { Ri :=R_i^{ij};
      j=j+1
      go to (6); }
  i=i+1;
}
```

(7) Until all $R_i \mid d=\Phi$
(8) End.

Notes: A d-associated tree will be produced by using this algorithm(Fig. 1). It can be bidirectional or unidirectional, depending on whether the dependency d is symmetric or anti-symmetric. Here we suppose d is anti-reflexive, anti-symmetric, but transitive. So the d-associated tree is unidirectional(Some arrows are upward and others are downward).

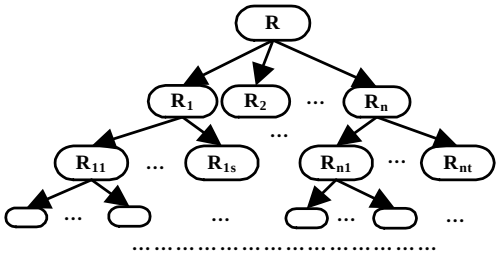

Figure 1. Associated Graph of a Dependency

B. Some Concepts about Ripple Effects

We propose the following definitions of ripple-effects for requirements evolution: The impacts of making changes to requirements sets. In order to define ripple width and ripple depth of a dependency, we first define Out-degree and In-degree of evolution of a node in a d-associated tree.

Definition 4.1 The number of the one-way edges that are related to a node R_i is called evolution Out-degree of R_i; The number of the nodes on the path from root to R_i, but except R_i is called evolution In-degree of R_i.

Definition 4.2 The ripple width and depth of a dependency are as follows:

In any subtree of a d-associated tree, supposing R is the root of this subtree, but not a leaf, has other n different nodes: $R_1, R_2,\ldots, R_n$; $s_1, s_2, s_3, \ldots, s_k$ ($s_i \geq 1$, i=1, …, k, $k \leq n$) are respectively the frequency of k different nodes in n. But

© 2012 ACADEMY PUBLISHER

because we suppose d is anti-reflexive and anti-symmetric, so we can judeg s_i=1, for all i=1,2,…,k. When R changes, it will probably affect other n requirements(nodes), we call wth=m$-\sum_{i=1}^{k}(s_i-1)$ as ripple width of changing R, here wth=m, because all s_i=1, i=1,2,…,k; m is the total Out-degree of all nodes in this subtree, the layer of this subtree is called ripple depth of changing R, recorded as lth.

Ripple width shows when a requirement changes, how many other requirements will probably be affected by it. Out-degree of a requirement does the same. In-degree of R shows how many requirements when they change will probably affect R.

Definition 4.3 Direct ripple degree and indirect ripple degree are defined as follows:

In any d-associated tree, any node R if it is not a leaf, has an Out-degree that is called as direct ripple degree of R (only one time for any repeated node), recorded as dth; wth-dth is called indirect ripple degree of R, recorded as idth.

C. The Formulas to Calculate the Ripple Effects of Requirements Changes

Now what follows is the research regarding the evolution ripple effects from a quantitative aspect.

(a) Adjacent matrix and reachability matrix

Definition 4.4 (Adjacent Matrix) Given $\{R_1, R_2,\ldots,R_n\}$ is the nodes of a d-associated tree, according to dependency d, adjacent matrix Md=(mij)m×n is defined as follows:

$$M_d=(m_{ij})_{m\times n}, \text{ here } m_{ij}=\begin{cases}1, & R_i dR_j \ or \ R_j dR_i,\\ 0, & R_i \overline{d} R_j.\end{cases}$$

It is apparently that all elements of main diagonal of M_d are zero.

Definition 4.5 (Reachability Matrix) The adjacent matrix M_d is called a one-step evolution reachability matrix, M_d^2 a two- step evolution reachability matrix, M_d^3 a three-step evolution reachability matrix, and so on. The transferable closure $M_d^+ = M_d \vee M_d^2 \vee M_d^3$ ……is called an evolution reachability matrix.

(b) Analyzing the ripple effects of requirements evolution

According to the structures of adjacent matrix and reachability matrix, we observe and present some properties about ripple effects of requirement evolution.

Property 4.1 In M_d, the number of 1 of each row is a direct ripple degree, corresponds to the requirement of this row, that is a one-step direct ripple degree; In M_d^2, the number of 1 of each row is a direct ripple degree, corresponds to the requirement of this row, that is a two-step direct ripple degree, and so on. In M_d^+, the number of 1 of each row is ripple width, correspond to the requireement of this row. In $M_d^+ - M_d$, the number of 1 of each row is an indirect ripple degree, corresponds to the requirement of this row.

Property 4.2 Given i≥1, in M_d^i, if existing a node R_j(j=1, 2, 3, …, n), its correspondent master matrix is zero, then the ripple depth of R_j is i-1.

When there are a large number of requirements in a project, it will show a benefit of this way to define direct ripple degree and indirect ripple degree. In a complex or big project, usually the dependencies among requirements are very complicated, and these two kinds of matrix will be very big, making it difficult to do matrix operations. To simplify matrix operations, we can focus on taking a master matrix of Ri into account, then easily calculate the ripple effects of Ri. In addition, the calculation of ripple effects is done by matrix operations, which makes it easy to confirm the propagation of changing requirements.

Property 4.3 In M_d^+, the number of 1 of each column is the number of its ancestors, correspondent to the requirement of this column, that is its In-degree of this requirement.

D. Rules of Controlling Ripple Effects

If not degrading a software to maintain its requirements specification, we need to consider all factors related to requirement evolution, to modify this requirements specification from the perspective of cost-effectiveness[7] and low side effect and other risks.

To control the ripple-effect of changing requirements, some rules are suggested as follows.

(1) Change requirements only if essential;

(2) When a requirement changes or is affected by other changing requirements, we should analyze other requirements possibly affected by it. We should use identification algorithms of requirement dependencies to confirm the extent and scope of the requirement that will be affected. We use these analyses to make decisions for change requests.

(3) If a change request comes, some changes of requirments need to be implemented, and the steps can be taken are:

1) Classify the types of changes, prioritize the changes, and confirm the difficult degree of implementing a requirement. Then, make decision to order the sequence of implementing changes;

2) Call the identification algorithms to confirm the ripple-effects of this changed requirement;

3) Repeating steps from 1) to 2) until all changes have to be implemented.

(4) Before implementing changes, alternatives should be decided. For each changing requirement, before and after it is changed, its difficult degree should be confirmed.

Therefore, in requirements management, requirements specifications should be effectively maintained, and as few as changes as possible should be made to change requirements, minimizing the impacts of changes.

© 2012 ACADEMY PUBLISHER

V. AN EXAMPLE

This example is a typical one in structured analysis of M.Jackson[26]. Here we need to design a list of fund declaration of a university, its format is following as Fig. 2.

A List Of Fund Declaration

Name	Number	Level	Money

Figure 2. Format of the List.

Here, Name is the name of the related fund. Level has three alternatives: city, province, and state, means the fund will be supported by the level of city, or province or state in China. Money is greater than or equal to 0. By analyzing the problem, 9 requirements can be produced(Fig. 3). From the

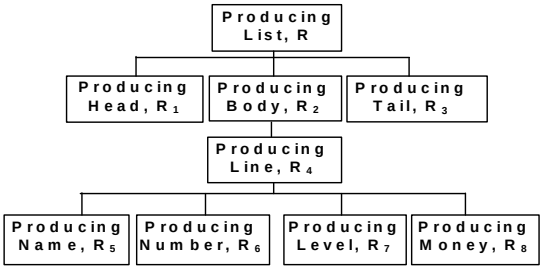

Figure 3. 9 Requirements

view of implementation[24], by *Algorithm 4.1* we can know R depends on R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, that is { R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8} $\rightarrow$ R. Also, { R_4, R_5, R_6, R_7, R_8} $\rightarrow$ R_2, $R_4 \rightarrow$ { R_5, R_6, R_7, R_8} (Only after producing lines, then can produce R_5, R_6, R_7, R_8). The tree of dependencies made for these 9 requirements is following as Fig .4.

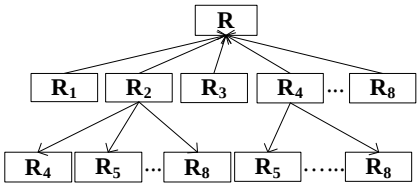

Figure 4. The Tree of Dependencies

Then we can get an adjacent matrix M_d and a reachability matrix M_d^+.

According to *Property 4.1*, in M_d, we can see that the direct ripple degrees of R_1, R_2, R_3, R_4, R_5, R_6, R_7 ,R_8 are respectively 1, 6, 1, 5, 1, 1, 1, 1. The change of R_2 will directly affect R_1, R_4, R_5, R_6, R_7, R_8. The change of R_4 will directly affect R, R_5, R_6, R_7, R_8. As $M_d - M_d^+$ is a zero matrix, means none of the indirect ripple degrees exists. In fact, this is easily shown by Fig. 3. We also can get some other quantities about ripple effects by M_d, M_d^2,…,M_d^+ according to *Property 4.2* and *Property 4.3*.

$$M_d = \begin{pmatrix} 0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&1&1&1&1&1 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&1&1&1&1 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \end{pmatrix},$$

$$M_d^2 = \begin{pmatrix} 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \end{pmatrix},$$

$$M_d^+ = \begin{pmatrix} 0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&1&1&1&1&1 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&1&1&1&1 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \end{pmatrix} = M_d .$$

Moreover, in M_d, there are many zero, so we can use the technique of block matrix theory to quickly calculate these quantities about ripple effects.

VI. CONCLUSION AND FUTURE WORK

Current research about requirements engineering mostly do not adequately connect theory and practice. More research is needed to develop new methodologies to study requirements engineering. However, if a method is too abstract, it will not be practicable, and will not be able to solve real problems in requirements engineering. It is preferable to build a methodology from quantitative perspective to study the phenomena in requirements engineering.

In this paper, we present a generic algorithm of requirements dependencies. We then use it to adopt matrix theory from quantitative perspective to investigate the ripple effects of requirements evolution, and we present some

© 2012 ACADEMY PUBLISHER

formulas and properties. The example has showed the effectiveness of this method. Furthermore, we discuss some rules of controlling ripple effects from four aspects. Our new understanding of requirements evolution's ripple-effect provides a basis and a new idea in dealing with the management of requirements changes in software development.

Future research includes:

It is necessary to establish identification algorithms for dependencies[25], and to integrate their properties of relational algebra such as reflexivity, symmetry, and transitivity. This will provide a better way to evaluate the ripple effects more accurately. Different dependencies will produce different ripple effects for changing requirements, so many types of dependencies should be considered at the same time for every requirement when it changes. Moreover, the ripple effects of requirements models can be investigated by the same approach.

ACKNOWLEDGMENT

The authors would like to thank Prof. Dr. Kassem Saleh and the reviewers for their valuable contributions to this paper. This work was supported in part by a grant from the Natural Science Foundation of Guangdong (101510631-01000046) and the National Science Foundation of China (60940033).

REFERENCES

[1] http://www.re11.org/, Call for Papers of Research Papers of 19th IEEE INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE.

[2] Yan Yu-qing, Li Shi-xian, Mei Xiao-yong.Analysis of Defect Requirements and Management Model[J].Computer Science, 2009,36(4):140-144.(in Chinese)

[3] Yan Yu-Qing, Li Shi-Xian, Liu Xian-Ming. Quantitative Analysis for Requirements Evolution's Ripple-Effect[C]. Bangkok,Thailand: IEEE Computer Society,Los Alamitos, CA,US, 2009.

[4] Massimo Felici. Taxonomy of Evolution and Dependability. In Proceedings of the Second International Workshop on Unanticipated Software Evolution[J], USE 2003, Warsaw, Poland, 5-6 April 2003, pp. 95-104.

[5] Yau,S.S.,Collofello,J.S.,MacGregor,T.Ripple Effect Analysis for Software Maintenance, Proc. COMPSAC-78, IEEE Computer Society, 1978, pp. 60 - 65.

[6] R.S. Arnold and S.A. Bohner.Impact Analysis-Towards a Framework for Comparison.Proceedings of the Conference on Software Maintenance Montreal, Que, 1993, pp. 292-301.

[7] Byron J. Williams Jeffrey Carver Ray Vaughn. Change Risk Assessment: Understanding Risks Involved in Changing Software Requirements. SERP'06 - The 2006 International Conference on Software Engineering Research and Practice.

[8] Haney, F. M. Module connection analysis: a tool for scheduling software debugging activities. Proceedings. AFIPS Joint Computer Conference. pp. 173-179. 1972.

[9] J.P. Queille, J.F. Voidrot, N. Wilde, and M. Munro, The Impact Analysis Task in Software Maintenance: A Model and a Case Study. Proceedings of the International Conference on Software Maintenance, Victoria, BC 1994, pp. 234-242.

[10] M.R. Strens, R.C. Sugden, Change Analysis: A Step towards Meeting the Challenge of Changing Requirements. IEEE Symposium and Workshop on Engineering of Computer Based Systems (ECBS'96), pp.278-283,1996.DOI: 10.1109/ECBS.1996.494539.

[11] Bashar Nuseibeh, Steve M. Easterbrook: The Process of Inconsistency Management: A Framework for Understanding. DEXA Workshop 1999: 364-368.

[12] A. Hunter and B. Nuseibeh. Analyzing inconsistent specifications. In Proceedings of the Third IEEE International Symposium on Requirements Engineering, pag-es: 78-86, 1997.

[13] Nuseibeh,B., Russo,A.M., Using Abduction to Evolve Inconsistent Requirements Specifications[J]. Australian Information Systems Journal, 1999, Vol: 7, Pages: 118 - 130.

[14] Finkelstein A.,Gabbay D.,Hunter A., Kramer,J., Nuseibeh B.,Inconsistency Handling In Multi-Perspective Specifications[J]. IEEE Transactions on Software Engineering, Vol. 20, No. 8, 1994, 569-578.

[15] Hunter A. , Nuseibeh B . Managing inconsistent specification : Reasoning, analysis and action[J]. ACM Trans . on Software Engineering and Methodology , 1998 , 7(4) : 335−367.

[16] S. Easterbrook and B. Nuseibeh. Using ViewPoints for Inconsistency Management. BCS/IEE Software Eng. J., vol. 11, no. 1, pp. 31-43, 1996.

[17] Talha Javed, Manzil-e-Maqsood and Qaiser S. Durrani. A Study to Investigate the Impact of Requirements Instability on Software Defects. ACM Software Engineering Notes, Volume 29 Number 4, May 2004:1-7.

[18] Chang J. and Richardson D.G, Static and dynamic specification slicing. In Proceedings of the fourth Irvine Software Symposium, April 1994.

[19] Ryder B. G. and Tip F.. Change impact analysis for object oriented programs. In ACMSIGPLAN- SIGSOFT workshop on Program analysis for software tools and engineering,. ACM Press, 2001. pp. 46-53.

[20] Hassine J., Rilling, J., Hewitt J., Dssouli R.. Change impact analysis for requirement evolution using use case maps[C]. The 8th International Workshop on Principles of Software Evolution,2005. pp:81-90.

[21] Åsa G. Dahlstedt, Anne Persson. Requirements Interdependencies- Moulding the State of Research into a Research Agenda[C]. The Ninth International Workshop on Requirements Engineering:Foundation for Software Quality (REFSQ 2003), held in conjunction with CAiSE 2003.

[22] Klaus Pohl. Process-Centered Requirements Engineering[M]. New York, NY, USA: John Wiley & Sons Inc.,1996.

[23] P. Carlshamre, K. Sandahl, M. Lindvallet al. An Industrial Survey of Requirements Interdependencies in Software Product Release Planning[C]. Fifth International Symposium on Requirements Engineering. Toronto,Canada. 2001. Digital Object Identifier: 10.1109/ISRE.2001.948547.

[24] Yan Yu-qing. The Study of the Ripple-effect of Requirements Evolution and Dependency. The thesis of P.h.D(unpublished).

[25] Stevens. W., G. Mcyers. and L. Constantine. Structured Design. IBM System Journal. Volume 13. Number 2, 1974. Digital Object Identifier: 10.1147/sj.132.0115

[26] Qian Le-qiu, Zhao Wen-yun, Niu Jun-yu. Software Engineering. Beijing: Press of Tsing Hua University. 2007.

© 2012 ACADEMY PUBLISHER

Yuqing Yan was born in Zhanjiang city, Guangdong Province, China in 1963. She received her Bachelor degree in mathematics from South China Normal University in 1985, and Master degree in computer software and theory from Sun Yatsen University in 1999. Currently she is studying for a PHD in Sun Yatsen University. She is an Associate Professor in Cisco School Informatics, Guangdong University of Foreign Studies, Guangzhou, China. She has published 15 papers such as Analysis of Defect Requirements and Management Model[J]. Computer Science, 2009,36(4):140-144(Chinese Edition). Her interests include software engineering, requirements management, operational theory, logic theory etc.

Shixian Li was born in Yudu, Jiangxi Province, China in 1944. He is a Professor and Doctoral Supervisor at Sun Yat-Sen University. His research interests include software engineering, formal semantics etc. He has directed and completed several scientific projects supported by the Natural Science Foundation of Gua-ngdong and the National Science Foundation of China.He published almost one hundred papers and 20 books. He translated the book of Barry W. Boehm's Software Engineering Ecomomics into Chinese and published it in 2004.

Weijun Sun was born in Anhui Province, China in 1975. He received his degree of Master in Computer Application in 2003 at Nanjing University of Aeronautics and Astronautics, China. Currently he is working on his doctoral degree in computer software and theory at Sun Yatsun University,China. He is a Docent in Faculty of Computer at Guangdong University of Technology, Guangzhou, China. He has published at least 6 articles. One of them is An Approach for Reverse Engineering of Web Applications[C]. 2008 International Symposium on Information Science and Engineering (ISISE 2008).Shanghai, China: IEEE, 2008: 98-102. (EI: 200913119-78065). His interest is in studying softw are engineering, model driven architecture, model evolution.

© 2012 ACADEMY PUBLISHER