Integrated Change Impact Analysis Approach for the Software Development Phase

Nazri Kama

Advanced Informatics School, Universiti Teknologi Malaysia, Malaysia
nazrikama@ic.utm.my

Abstract

Impact analysis aims to quickly predict the extent of changes which will be required to software project products as a result of a potential change request. The existence of partially developed classes in the software development phase artifacts causes several challenges to the current static analysis and dynamic analysis techniques. Thus, this paper proposes a new approach that integrates between these two techniques to support those challenges. There are two main contributions of this paper: (1) a new change impact analysis approach that is able to support the software development phase; and (2) evaluation results that show the new approach gives higher accuracy of results over the selected current impact analysis approaches.

Keywords: *Impact analysis, **s**oftware development phase, traceability*

1. Introduction

Software is the most malleable component of a system [1]. It undergoes changes not only in the requirement phase, but also throughout the software development phases [2]. According to Bohner, changeability is one of the essential difficulties in software development [1].

It is important to manage the changes in the software to meet the evolving needs of the customer and hence, satisfy them [1]. Accepting too many changes causes delay in the completion and it incurs additional cost. Rejecting the changes may cause dissatisfaction to the customers. Thus, it is important for the software project manager to make effective decisions when managing the changes during software development. One type of information that helps to make the decision is the prediction of the number of classes affected by the changes. This prediction can be done by performing change impact analysis [1, 2].

Current impact analysis techniques include static analysis techniques [3, 4] and dynamic analysis techniques [5, 6]. These techniques are mainly developed for the software maintenance phase. The implementation of these techniques is based on the assumptions that: (1) all classes in the class artifacts are completely developed; and (2) the class artifact is used as a source of analysis since it represents the final forms of user requirements [7]. Unfortunately, these assumptions are not practical for implementing impact analysis in the software development phase since some classes in the class artifacts are still under development or partially developed [8].

The existence of partially developed classes in the class artifacts causes several problems to these static analysis and dynamic analysis techniques. The static analysis technique faces a problem related to the accuracy of program static information (i.e., class interactions) that is generated from source code through reverse engineering. The generated class interactions that involve partially developed classes may not represent the actual class interactions as some of the interactions have not been developed yet. On the other hand the dynamic analysis

techniques [5, 6] tend to produce inaccurate method execution paths that are generated from source code through reverse engineering. This is because some method execution paths that involve partially developed classes may have not been developed yet. The inaccuracy of the generated program static information from the static analysis technique and method execution paths from the dynamic analysis technique indirectly lead to inaccuracy of impact analysis results.

This paper proposes a new approach that is able to perform impact analysis during software development phase. Our approach combines between static and dynamic analysis techniques. To overcome the current static analysis technique problem, the approach develops the program static information (i.e., class interaction prediction) using a predictive technique by analyzing the requirement artifacts and the design artifacts instead of using reverse engineering process from the source code. For the dynamic analysis technique problem, the approach introduces partially developed class analysis as part of its process. This process is used to modify the generated method execution paths from the reverse engineering process. The modification is meant to improve the accuracy of the generated method execution paths.

Due to space limitation, this paper will not describe all processes in details in the approach. Basically there are two main stages in the approach: (1) Stage 1: Developing class interactions prediction. Detailed of this stage implementation can be referred to [9-13] and; (2) Stage 2: Performing impact analysis whereas detailed of this stage implementation can be referred to [14]. However, to give a clear picture of the overall implementation of the approach, we briefly describe all processes in general.

This paper is laid out as follows: Section 2 justifies related work. Next, Section 3 describes the overall structure of the new approach. Then, Section 4 gives evaluation strategy and followed by Section 5 that describes the evaluation results and analysis. Thereafter, Section 6 explains conclusion and future work.

2. Related Work

There are two categories of impact analysis techniques which are the static analysis technique and the dynamic analysis technique. The following sub-sections briefly describe the impact analysis definition and current most related works from each impact analysis category.

2.1 Impact Analysis

One of the most referred definitions of impact analysis is "a process of identifying potential consequences of a change, or estimating what needs to be modified to accomplish a change [1]".The motivation behind the impact analysis activity is to identify software artifacts (*i.e.*, requirement, design, class and test artifacts) that are potentially to be affected by a change. The change can be in a form of addition, removal and modification of new or existing software artifacts. With information on potentially affected software artifacts, effective planning can be made on what action will be undertaken with respect to the change.

There are two categories of impact analysis techniques [10, 11] which are the static analysis technique and the dynamic analysis technique. The static analysis technique develops a set of potential impacted classes by analyzing program static information that is generated from software artifacts (*i.e.*, requirement, design, class and test artifacts). Conversely for the dynamic analysis technique, this technique develops a set of potential impacted classes by analyzing program dynamic information or executing code. The following sub-sections describe each impact analysis category.

2.1. Static Analysis Technique

There are two most related current static analysis techniques to the new proposed approach which are the Use Case Maps (UCM) technique [5-6] and the class interactions prediction with impact prediction filters (CIP-IPF) technique [9,10].

The UCM technique [3] performs impact analysis on the functional requirements and the high level design model. This technique assumes that all the functional requirements and the high level design model are completely developed. This technique has two limitations which are: (1) there is no traceability technique used from the functional requirements and the high level design artifacts to the actual source codes. This technique only makes an assumption that the content of these two artifacts that is represented using the UCM model are reflected to the class artifacts. Any affected elements in the UCM model are indirectly reflected to the affected class artifacts; and (2) there is no dynamic analysis or source code analysis involved in this technique. Based on the precept that some of the effect of a change from a class to other class(es) may only be visible through dynamic or behavior analysis of the changed class [15-16], results from this technique tend to miss some actual impacted classes.

Next, the CIP-IPF technique [9-10] uses a class interactions prediction as a model for detecting impacted classes. This technique has its strength compared to the UCM technique. Comparing to the UCM technique, this technique has traceability link detection between the requirements artifacts and the class artifacts feature. This feature is used to navigate impact of changes at the requirement level to the class artifacts.

2.2. Dynamic Analysis Technique

There are two techniques which are the Influence Mechanism technique [7] and the Path Impact technique [8]. Basically, these techniques predict the impact set (classes or methods) based on method level analysis.

The Influence Mechanism technique [7] introduces the Influence Graph (IG) as a model to identify impacted classes. This technique uses the class artifacts as a source of analysis and assumes that the class artifacts are completely developed. There is a limitation for this technique which is there is no formal mapping or traceability process from requirements artifacts or design artifacts to class artifacts. This process is important in impact analysis process as changes not only come from class artifacts but it also comes from design and/or requirements artifacts. Since design and requirements artifacts do interact among them vertically (between two different artifacts of a same type) and horizontally (between requirement and design artifacts), changes that happen to them could contribute to different affected class artifacts. In some circumstances, focusing on the source code analysis may not able to detect those affected classes.

The Path Impact technique [8] uses the Whole Path DAG (directed Acyclic Graph) model [17] as a model to identify impacted classes. The concept of implementation for this technique is almost similar to the Influence Mechanism technique as this technique uses the class artifacts as a source of analysis and assumes that the class artifacts are completely developed. Also, this technique performs a preliminary analysis prior to performing a detail analysis. There are two limitations of this technique. First, the implementation is time consuming as the technique opens to a huge number of data when the analysis goes to a large application. Next, there is no formal mapping process from requirements artifacts or design artifacts to class artifacts. As described earlier, this process is important in impact analysis process as changes not only come from class artifacts but also from design and/or requirements artifacts.

3. Integrated Approach

Three main elements are considered by the integrated approach (or the "new approach") to support impact analysis in the software development phase. The elements are: developing class interactions prediction (as the impact analysis model) using a new predictive technique; including the partially developed class analysis in the dynamic analysis technique implementation; and implementing the static analysis technique as well as the dynamic analysis technique when necessary.

The first element is the developing class interactions prediction using a new predictive technique. The predictive technique develops class interactions prediction through reflection of significant object interactions in the requirement artifact via horizontal traceability links. The significant object is an object that has potential to be used as a class name in actual class implementation. Thus, the significant object is an object that has a relationship with the class that has a similar name with the significant object name. The relationship is also called "traceability links" [1]. The reflection process is based on the precept that the significant object interactions in the requirement artifact reflect to the class interactions in the class artifact.

The second element is the inclusion of the partially developed class analysis in the dynamic analysis technique implementation. Typically the existing dynamic analysis in particular to the method-based analysis [6, 15, 16] consists of two main steps which are generating method execution paths, and detecting a set of actual impacted classes based on the generated method execution paths. The first step generates the method execution paths from a set of classes in the class artifact using a path generator tool whereby the second step analyses the generated method execution paths to detect a set of actual impacted classes according to change requests.

We take a closer look at the first step in the existing dynamic analysis steps. This step uses a set of classes that are assumed to be completely developed to generate the method execution paths. Unfortunately from the software development phase perspective, the generated method execution paths tend to be inaccurate as some interactions between methods that involve the partially developed class may have not been developed yet. Thus, it will indirectly affect the impact analysis results. Therefore, this approach includes the partially developed class analysis in its process to improve the accuracy of the method execution paths.

The third element is the implementation of the static analysis technique as well as the dynamic analysis technique when necessary. Performing impact analysis using the static analysis and the dynamic analysis independently has it owns problems that affects the accuracy of the results. As described earlier, the static analysis technique faces a problem related to the accuracy of program static information (i.e., class interactions) that is used as a source of analysis. On the other hand, the dynamic analysis technique faces a problem related to its assumption on the status of the class artifact in which they assume that all classes in the artifact are fully developed.

This approach implements the static analysis technique as well as the dynamic analysis technique when necessary. That is, the dynamic analysis will only be implemented if there exists some partially developed classes in the initial set of impacted class artifacts. Otherwise, only the static analysis technique will be implemented. Figure 1 below shows the stages of the approach.

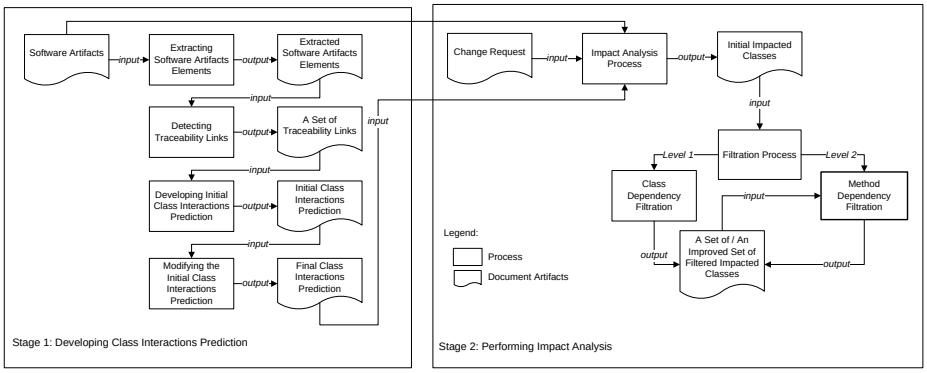

Figure 1. Overall impact analysis approach

3.1 Stage 1: Developing Class Interactions Prediction

This approach uses a predictive technique to develop class interactions prediction model. In brief, the new predictive technique develops the class interactions prediction based on two analyses which are: (1) significant object interactions analysis in the requirement artifact; (2) design patterns analysis in the design artifact. The first analysis analyses the significant object interactions in the requirement artifact to develop an initial class interactions prediction via horizontal traceability links. For the horizontal traceability links, the new predictive technique refines the selected current technique which is the Rule-based technique [20, 21].

The difference between the Rule-based technique [20, 21] and the refined technique is that rule-based technique performs a similarity analysis between the requirement statement and the analysis object model elements whereby the refined technique performs a similarity analysis between the significant object names and the design class properties names (*i.e.*, class name or class attribute name). The similarity analysis may not discover sufficient conditions for satisfiability. However, this will be improved in the future work.

The second analysis is the design patterns analysis. This analysis is considered as an important analysis for the new predictive technique because the current techniques [3, 4] do not exploit the design artifacts. This analysis modifies the initial class interactions prediction produced by the first analysis according to design patterns. At this moment, this stage performs the analysis according to the Boundary-Controller-Entity (BCE) design pattern only. However, the developed steps for this analysis are flexible in that it can also be used to implement other design patterns analyses.

There are four processes in this stage. The first process is the extracting software artifact elements. There are three types of software artifact involved in this process which are requirement artifact, design artifact, and class artifact.

The second process is the detecting traceability link. There are two categories of traceability link which are the horizontal traceability link and the vertical traceability link. For the horizontal traceability links, there are two types of links which are from requirements in the requirement artifact to design class in the design artifact links and from design class in the design artifact to class in the class artifact links. On the other hand the vertical traceability links detects traceability links among requirements in the requirement artifact (or requirement interactions) and the vertical traceability links among design classes in the design artifact (or design class interactions).

The third process is the developing initial class interactions prediction. This process concentrates on developing an initial set of class interactions prediction. This set is developed based on the reflection of significant object interactions that exist in the requirement artifact.

The last process is the modifying the initial class interactions prediction. This process modifies the initial class interactions predictions according to the selected design patterns that have been used by the system or application.

3.2 Stage 2: Performing Impact Analysis

In brief, the new impact analysis technique consists in performing the static analysis technique as well as the dynamic analysis technique when necessary, i.e. when partially developed classes exist in the initial set of potential impacted classes, the new approach performs the dynamic analysis technique as well as the static analysis technique. The implementation of the dynamic analysis technique aims at improving the initial set of potential impacted classes produced by the static analysis technique. Conversely when there are no partially developed classes exists in the initial set of potential impacted classes, only the static analysis technique is performed.

Also, the new impact analysis technique provides high level steps that support the inclusion of the partially developed class analysis in the dynamic analysis technique implementation. The steps are designed to be independent from any specific instance of dynamic analysis technique implementation. At this moment, the approach uses the Path-Impact [6] technique as an instance of the dynamic analysis technique implementation.

This stage identifies a set of potential impacted classes using the class interactions prediction (Stage 1 result) according to change requests. There are two main processes in this stage which are the impact analysis process and filtration process.

The impact analysis process identifies a set of potential impacted classes using the class interactions prediction according to change request specification. The outcome of this process is a set of initial impacted classes.

The filtration process aims at eliminating some typically false results generated by the impact analysis process. There are two filtration levels in this process which are the Class Dependency Filtration (CDF) level and the Method Dependency Filtration (MDF) level. The CDF level implements the static analysis on the initial set of potential impacted classes produced by the impact analysis process. This implementation is important by the fact that some interaction links in the initial set of potential impacted classes have no change impact value. The interaction link that has no change impact value means that if change happens to one side of two interacting classes, the other class will not be affected. This is because the other class does not require the changed class for its implementation.

The MDF level implements the dynamic analysis on a set of method execution paths generated from the set of filtered impacted classes produced by the CDF level. This implementation is important by the fact that in some situations the CDF level filtration tends to include false impacted class in the filtered set of impacted classes. This situation exists when larger classes where a small class only uses part of the services offered. In other words, change happens to a particular method in the larger class does not affect the method that the small class uses. For example, three methods exist which are a method (M3) in C1 and two methods (M1 and M2) in C2. Two scenarios of interaction between the methods are: (1) M3 requires M2 for its implementation and; (2) M2 has no dependency from M1. When change happens to M1 in C2, it will not affect C1 even though C1 requires C2 for its implementation. This constitutes the main challenge in the CIP-IPF technique implementation [9, 11].

4. Evaluation Strategy

The goal of this evaluation is to answer a question of “does the new approach give better accuracy of impact analysis results than the selected current impact analysis techniques?

There are two most related current impact analysis approaches have been selected which are: (1) the Class Interactions Prediction with Impact Prediction Filters (CIP-IPF) approach; and (2) the Path Impact approach. There are many evaluation strategies were constructed and among the important strategies are: (1) subject and case study; (2) software development process; (3) evaluation procedure; (4) evaluation metrics; (5) hypothesis; and (6) data analysis

4.1 Subject and Case Study

The subjects of the experiment were three groups of final year post-graduate students of software engineering course at Advanced Informatics School, Universiti Teknologi Malaysia (UTM). During their professional attachment session in the industry, we were involved as one of the software developers in these projects. We developed some of the modules which were then used as the case studies.

Three case studies or software development projects (P) were selected which are: (1) P1- Car Management System: a system that simulates a car driving assistant. The simulation covers some major car management functionalities such as starting the engine, managing auto-cruise mode, calculating average fuel consumption and monitoring car maintenance functions; (2) P2- Database Encryption System: is a tool that provides encryption and decryption facilities in the Oracle RDBMS. This system consists of two main modules which are key management, and encryption and decryption of data modules; and (3) Course Registration System: This system provides an automated course registration for the university's student. Among the functionalities provided by the system are the system allows the students to register courses through web, and the students are able to access the system to view their report card.

4.2 Software Development Process

There are four typical software development phases in the software development process . They are the requirement phase, design phase, coding phase and testing phase [17]. The selection of the software development process is crucial as the new impact analysis approach is meant to be used for the software development phase. In this experiment we select the Waterfall software development model [17] as this model is considered as the basic process model. Besides that, the case study that is used in this experiment is considered as a small software development project. To note, we aware of the latest software development model like Agile model [17]. We will take this model implementation into consideration in our future work.

4.3 Evaluation Procedure

There are four steps in the evaluation procedure.

4.3.1 Step 1: Issuing fifteen change request specifications to all software projects: for the purpose of performing the impact analysis evaluation, we issued a set of change requests to the developed modules and identified the impact analysis results according to the issued change requests.

4.3.2 Step 2: Developing a set of potential impacted classes according to the change request specifications: The subjects in each software project are required to develop three sets of potential impacted classes for the issued change requests using the three impact analysis approaches. Prior to developing the three sets of results, the subjects were given a briefing session on how to use the three impact analysis techniques. To ensure that all

subjects have sufficient understanding on these techniques' implementation, the session is supported by an extensive examples and guidelines from literature.

4.3.3 Step 3: Developing actual set of impacted classes according to each change request specification: The subjects in each software project are required to develop actual set of impacted classes through actual change implementation. The subject starts by analyzing the issued change request to identify the primary and secondary impacted requirements. After that, based on the detected impacted requirements, the primary impacted classes are then identified manually based on the subject's experience in developing the case study. Then, based on the identified primary impacted classes, the secondary impacted classes are identified through actual change implementation. After the actual implementation, all the detected primary and secondary impacted classes are then recorded.

4.3.4 Step 4: Measuring accuracy of the set of potential impacted classes that are generated by the new approach compared to the selected current impact analysis approaches: The detected primary and secondary impacted classes produced by all techniques in Step 2 will be compared to the actual impacted classes produced in Step 3. The comparison uses a set of evaluation metrics that will be described in the following section.

4.4 Evaluation Metrics

This study employed evaluation metrics as described in [19]. Briefly, each impacted class predictions were categorized according to four numbers: (1) Not Predicting and Not Changing (NP-NC): number of pairs of classes correctly predicted to not be changing; (2) Predicting and Not Changing (P-NC): number of pairs incorrectly predicted to be changing; (3) Not Predicting and Changing (NP-C): number of classes incorrectly predicted to not be changing; and (4) Predicting and Changing (P-C): number of classes correctly predicted to be changing. These numbers are then used to calculate a kappa value [20]. This value reflects the accuracy or the prediction (0 is no better than random chance, 0.4-0.6 is moderate agreement, 0.6-0.8 is substantial agreement, and 0.8-1 is almost perfect agreement).

4.5 Hypotheses

A hypothesis that investigates the effectiveness of the new approach is developed. The new approach represents a combination of the static analysis and dynamic analysis techniques whereby the selected current impact analysis approaches represent an independent analysis approach (CIP-IPF- static analysis approach only; Path Impact- dynamic analysis approach only). If the combination is not effective, H_0 is accepted: *H_0: The new approach does not give higher accuracy of impact analysis results than the selected current approaches results. H_a: The new approach gives higher accuracy of impact analysis results than the selected current approaches results*

4.6 Data Analysis

To validate the hypothesis, the Independent T-Test statistical analysis was used. Two stages of analysis are created. The first stage compares Means results between the CIP-IPF approach and the new approach whereas the second stage compares Means results between the Path Impact technique and the new approach.

This analysis determines whether there is a statistically significant difference between the means of two unrelated group of data (independent and dependent data) or not in both tables. The null hypothesis for the Independent T-Test was that the population means from the two

independent and dependent data in each table are equal, H_0: d1 = d2. The alternative hypothesis for the Independent T-Test was that the population means from the two independent and dependent data in each table are not equal, H_0: d1 ≠ d2. To do this, a significance level (alpha) that is used to either reject or accept the hypotheses is set to 0.05. To identify whether the SDP-CIA results improves the selected current impact analysis approaches (CIP-IPF approach [9], the Path Impact approach [7]) or not, a comparison of mean values produced by the SDP-CIA to the CIP-IPF and Path Impact approaches were then conducted.

5. Evaluation Results and Analysis

Table 1 below shows the Independent T-Test results.

Table 1. Summary of Independent T-Test Results

	Technique	Means Results
Kappa Value	CIP-IPF Technique	0.7927
	The New Approach	0.9060
Kappa Value	Path Impact Technique	0.7773
	The New Approach	0.9060

5.1 Stage 1 Analysis: The CIP-IPF Technique vs. The New Approach

Looking at Table 1 results, the Mean results from both approaches show the new approach value is 0.9060 and the CIP-IPF approach value is 0.7927. This shows that the new approach value is higher than the CIP-IPF approach. Thus, the values reject the null hypothesis (H_0: The new approach does not improve on the CIP-IPF approach results) and accept the alternate hypothesis (H_a: The new approach gives higher accuracy of impact analysis results than the CIP-IPF approach).

5.2 Stage 2 Analysis: The CIP-IPF Technique vs. The New Approach

Looking at Table 1 results, the Mean values show the new approach value is 0.9060 and the Path Impact approach value is 0.7773. This shows that the CIP-IPF approach value is higher than the Path Impact approach. Thus, the values reject the null hypothesis (H_0: The new approach does not give higher accuracy of impact analysis results than the Path Impact approach) and accept the alternate hypothesis (H_a: The new approach gives higher accuracy of impact analysis results than the Path Impact approach).

6. Conclusion and Future Work

The main contribution of this paper is a new integrated approach between the static analysis technique as well as the dynamic analysis technique. This integration intends to overcome the current static analysis and dynamic analysis techniques challenges in order to support change impact analysis for the software development phase. Besides that, a demonstration on the effectiveness of the new approach is presented as well.

As for the future works, we increase the effectiveness of change management using the new approach in the software development phase, a set of guidelines that supports the technique's implementation in the current software development lifecycle/process needs be developed. The guidelines are expected to be easily-incorporated in any software development lifecycle/process. Also, we will extend the experiment by including a performance metric. This metric measures the amount of time that is used to identify a set of

potential impacted classes according to a particular change request. Prior to extending the experiment, the approach's implementation needs to be automated. The automation covers all Stage 1 and Stage 2 implementation processes. This automated process is expected to assist the experiment to get reasonable implementation times required by the approach for a particular change request.

Acknowledgements

We would like to thank to Universiti Teknologi Malaysia (UTM) and Ministry of Higher Education (MoHE) Malaysia for their financial support under the Exploratory Research Grant Scheme (ERGS) Vot No 4L022. Also to final year post-graduate students of software engineering course at the Advanced Informatics School, Universiti Teknologi Malaysia who have been involved in the study.

References

[1] S. Bohner and R. Arnold, "Impact Analysis - Towards a Framework for Comparison", Proceeding of the International Conference on Software Maintenance, **(1993)** September 27-30, Montreal, Canada.
[2] R. J. Turver and M. Munro, "An Early Impact Analysis Technique for Software Maintenance", Journal of Software Maintenance: Research and Practice, vol. 6, no. 1, **(1993)**.
[3] J. Hassine, J. Rilling, J. Hewitt and R. Dssouli, "Change Impact Analysis for Requirement Evolution Using Use Case Maps", Proceeding of the 8th International Workshop on Principles of Software Evolution, **(2005)** September 5, Washington, US.
[4] M. Shiri, J. Hassine and J. Rilling, "A Requirement Level Modification Analysis Support Framework", Proceeding of the 3rd International IEEE Workshop on Software Evolvability, **(2007)** October 1, Montreal, Canada.
[5] B. Breech, A. Danalis, S. Shindo, and L. Pollock, "Online Impact Analysis via Dynamic Compilation Technology", Proceeding of the 20th IEEE International Conference on Software Maintenance, **(2004)** September 11-17, Washington, US.
[6] J. Law and G. Rothermal, "Incremental Dynamic Impact Analysis for Evolving Software Systems", Proceeding of the 14th International Symposium on Software Reliability Engineering, **(2003)** November 17-20, Washington, US.
[7] K. H Bennet and V. T. Rajlich, "Software Maintenance and Evolution: A Roadmap", Proceeding of the International Conference on the Future of Sofware Engineering, **(2003)** May 3-10, New York, USA.
[8] N. Kama, T. French and M. Reynolds, "Predicting Class Interaction from Requirement Interaction", Proceeding of the 13th IASTED International Conference on Software Engineering and Application, **(2009)** December 2-5, Boston, US.
[9] N. Kama, T. French and M. Reynolds, "Predicting Class Interaction from Requirement Interaction: Evaluating A New Filtration Approach", Proceeding of the 13th IASTED International Conference on Software Engineering, **(2010)** April 6-8, Innsbruck, Austria.
[10] N. Kama, T. French and M. Reynolds, "Considering Patterns in Class Interactions Prediction", Advances in Software Engineering Book, vol. 117, **(2010)**.
[11] N. Kama, T. French and M. Reynolds, "Design Patterns Consideration in Class Interactions Prediction Development", International Journal of Advanced Science and Technology, vol. 28, no. 6, **(2011)**.
[12] N. Kama and F. Azli, "Requirement Level Impact Analysis with Impact Prediction Filter", Proceeding of the 4th International Conference on Software Technology and Engineering, **(2012)** September 1-2, Phuket Thailand.
[13] N. Kama, T. French and M. Reynolds, "Impact Analysis using Class Interactions Prediction Approach", Proceedings of the 9th New Trends in Software Methodologies, Tools and Techniques, **(2012)** September 29 – Oct 1, Yokohama, Japan.
[14] L. Huang and Y. T Seong, "Dynamic Impact Analysis Using Execution Profile Tracing", Proceeding of the 4th International Conference on Software Engineering Research, Management and Applications, **(2006)** August 9-11, Washington, USA.
[15] L. Huang and Y. T Seong. Precise Dynamic Impact Analysis with Dependency Analysis for Object-Oriented Programs", Proceeding of the 5th ACIS International Conference on Software Engineering Research, Management & Applications, **(2007)** August 20-22, Washington, USA.

[16] J. R. Larus, "Whole Program Paths", Proceeding of the ACM SIGPLAN Conference on Programming Language Design and Implementation, **(1999)** May 20-22, New York, US.
[17] I. Sommerville, Software Engineering (7th Edition), Pearson Education, **(2008)**.
[18] M. Lindvall and K. Sandahl, "How Well Do Experienced Software Developers Predict Software Changes", Journal of Systems and Software, vol. 43, no. 1, **(1998)**.
[19] J. Cohen, "A Coefficient of Agreement for Nominal Scales", Journal of Educational and Psychological Measurement, vol. 20, no. 1, **(1960)**.
[20] G. Antoniol, G. Canfora and G. Casazza, "Information Retrieval Models for Recovering Traceability Links between Source Code and Documentation. Proceedings of the International Conference on Software Maintenance, **(2000)** October 11-14, California, US.
[21] G. Spanoudakis, "Plausible and Adaptive Requirements Traceability Structures", Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, **(2002)** July 15-19, New York, US.

Author

Nazri Kama received his Master Degree in Real-time Software Engineering and Bachelor Degree in Management Information System from Universiti Teknologi Malaysia in 2002 and 2000 respectively. He then obtained his PhD degree at The University of Western Australia in 2011. His research interests are in software development, software maintenance, impact analysis, traceability, and requirement interactions.