A Survey to Determine the Significance of Software Requirements Defects from Practitioners' Perspectives in Malaysia

Sabrina Ahmad and Siti Azirah Asmai

Abstract— It is important to have quality software requirements as they set a foundation to the software project resource estimation, provide a baseline for architecture and design and eventually mould the shape of the software products. Besides, history proves that good quality software requirements are crucial and the cost of fixing defects increased exponentially the later the defects were detected in the software development life cycle. Through a survey technique, this paper presents a study to position the significance of software requirements defects in the software development life cycle. It was conducted among practitioners in Malaysia which includes various industry background and level of experiences. Based on the survey analysis, the IT practitioners agreed on the existence of software requirements defects in the current practice and the types of defects were identified. Referring to the early stage of requirements elicitation, the types of defects identified were incomplete, incorrect, inconsistent, unclear, infeasible, irrelevant and incomprehensible. This paper is expected to provide an insight into the relevance of conducting research to deal with software requirements defects.

Keywords— *correlation, quality requirement, requirements engineering, software requirements defects, survey*

I. INTRODUCTION

In the process of developing software products, defects are inevitable. This is especially so when the development involves various stages, processes and stakeholders. Literature shows that much effort has been put to further improve the prediction, the discovery, the management and the elimination of software defects. However, the attempt to handle defects in software requirements is inadequate. It is believed that the requirements defects are barely exist and will eventually resolve as the development progress throughout the system development life cycle. On the other hand, literatures and a survey conducted demonstrated otherwise. This paper will present the literatures and the findings on the existence of defects in software requirements in the industry in Malaysia.

This work is funded by Fundamental Research Grant Scheme numbered FRGS/2/2013/ICT01/FTMK/02/1/F00184 by the Ministry of Education Malaysia.

Sabrina Ahmad is with the Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, Durian Tunggal 76100 Melaka, Malaysia (e-mail: sabrinaahmad@utem.edu.my).

Siti Azirah Asmai is with the Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, Durian Tunggal 76100 Melaka, Malaysia (e-mail: azirah@utem.edu.my).

Why is it important to position the relevance of software requirements defects existence? It is important to instill awareness to both practitioners and researchers to keep working on improving the quality of software requirements. In fact, history proves that good quality requirements are crucial for the success of system development. Getting requirements right might be the single most important and difficult part of a software project. Considering the importance of software requirements, this paper presents an investigation through literatures and a survey among practitioners in Malaysia to determine the significant existence of software requirements defects. It also presents the attributes of requirements defects which differ from software defects. In relation to that, when the defects occurred and how the practitioners currently handling them are presented too.

Following Introduction, Section 2 presents the current effort by researchers in literatures. It is then followed by the explanation of the economic impact of software requirements defects in Section 3. Next, Section 4 explains the survey methodology. Section 5 presents the results and discussion. Finally, Section 6 concludes the paper.

II. LITERATURE REVIEW

A literature review has been done to discover current efforts on software requirements defects. The focus of the exploration is to position the existence of the defects and if it is worth conducting research to remove or at least to minimize the impact of the defects. The literature is developed based on a research question. The question is focus on to find the evidence on the relevance of software requirements defects to the practitioners and researchers and their limitations. The research question is:

"What is software requirements defect and does it exist?"

The motivation of the research question stated above is as below:

"Discovering the definition and types of software requirements defects, if exist."

A. Search Strategies and Search Selection

The search was done through digital libraries and databases; found through search string, and refining search string. Below is the list of the digital databases that being used to search papers in our study:

1) ACM Digital Library (dl.acm.org)
2) Cornell University Library (arXiv.org)
3) Elsevier (Elsevier.com)
4) Google Scholar (scholar.google.com)
5) IEEE Xplore (ieeexplore.iee.org)
6) ScienceDirect (sciencedirect.com)
7) Scopus (scopus.com)
8) Springer (Springerlink.com)

The search strings were based on the research questions and the keywords of the research field. It was limited by the year of 1996-2015 including journal papers and conference proceedings. Language for the search was limited to English only. The general keyword that we used in searching the related articles for the literature review were "systematic literature review", "software requirement engineering", "requirement engineering" and "requirements defects". Other than that we specified the search to "defect", "error", "fault", "software problem", "requirement elicitation phase", "requirement elicitation" and "defect classifications". The searched was resulting in the various reliable journals and conference proceedings covering issues in the software requirement defects.

Upon the completion of the searching process, we filtered the findings and narrow them to related works only. The next phase of review was based on the references of relevant papers to the research questions and the filtered papers were added to the final list for further analysis.

B. The Review

In a software development, defects usually referred to software defects which appear in the code of a software system. A software defect is an error, failure or fault in software [1] that produces an incorrect or unexpected result, or causes it to behave in unintended ways. It is a deficiency in a software product that causes it to perform unexpectedly [2]. Software defects are expensive in quality and cost. Moreover, the cost of capturing and correcting defects is one of the most expensive software development activities [3]. It is even worst when the defects are originated from requirements. Software requirements are critically important because defects originated in requirements engineering stage will propagate to the subsequent stages and therefore affect the entire software development process. Defects in requirements exist in the requirements statements usually presented in a requirements document.

A software requirements document is a formal documentation to record the requirements following the completion of requirements engineering process[4]. It also includes the identification of requirements and software and system requirements specifications. Eliciting requirements is crucial to determine the functionality and constrains of the system to be developed. Dealing with multiple stakeholders and sources, the interpretation of the requirements into a document has high potential to contain defects. It is especially so when the process involve human factors which is reported to contribute thirty-four of the risks in the research of Global Software Development [5]. IEEE defined defect as fault or an incorrect step, process or data definition in a computer program (IEEE, 1994). A defect is a problem that occurs in an artefact and may lead to a failure [6]. Any blemish, imperfection, or undesired statements in the product is defined as defect [7].

Definition of defects and its acronyms are stated below [8] and it is consistent with software engineering textbooks [9] and an IEEE standard [10]:

1) **Error** – defect in the human thought process made while trying to understand given information, solve problems, or to use methods and tools. In the context of software requirements specifications, an error is a basic misconception of the actual needs of a user or customer.
2) **Fault** – concrete manifestation of an error within the software. One error may cause several faults, and various errors may cause identical faults.
3) **Failure** – departure of the operational software system behaviour from user expected requirements. A particular failure may be caused by several faults and some faults may never cause a failure.

Defects are always being mentioned to be appearing in the line of codes. However, a study had discovered that defects found in software requirements are in higher percentage compared to the defects in the line of codes [11]. Besides, the effect of software requirements defects are more severe as the defects will propagate and affect the sequential stages of software development lifecycle. Therefore, the artefacts developed since then such as architecture, design and test cases were based on the wrong foundation.

Table I. List of requirements defects types.

Defect types	**Quality attributes**
Infeasibility	Feasibility
Imprecise	Precise
Incomplete	Complete
Winding	Concise
Incorrect	Correct
Misinterpret	Interpretable
Externally inconsistent	Externally consistent
Internally inconsistent	Internally consistent
Unmodifiable	Modifiable
Redundant	Not redundant
Unorganized	Organized
Unusable	Reusable
Ambiguous	Unambiguous
Incomprehensible	Understandable
Unnecessary	Necessary

Usually, requirements defects appear in requirements statements while transforming the various stakeholders' wish list gathered into requirements specification. There were two research works identified to work on software requirements quality attributes [12, 13]. The quality attributes if read in reverse will give a list of software requirements defects types as presented in Table I.

The list gives us an indication of qualities to achieve and at the same time the defects types that might exist in the requirements documents. This could be a fundamental guideline to express the difference between defects in requirements and software. However, the types of software defects are not included in this paper. Software defects are basically active in nature in which they can be detected and removed during software run time. Unlike requirements defects, the defects types are passive and need to be identified from the requirements documents.

Table II. Types of defects

Defects	Explanation
Missing or incomplete	The requirement is absent in document
Incorrect information	The information contained the false requirement
Inconsistent	The requirement is inconsistent with the overall document and in conflict with another requirement that is correctly specified
Ambiguous or unclear	Information or vocabulary have more than one interpretation
Misplaced	The requirement information is misplaced either in the section of the requirement specification document or in the functionalities, packages or system.
Infeasible	The requirement is not implementable
Redundant or duplicate	Requirement duplicate of another requirement or part of it already present in the document
Typo or formatting	Orthographic, semantic, grammatical error or missing word
Not relevant	Unnecessary information or out of project scope

A research has been done to classify software requirements defects [6] as stated in Table II. Based on the definition given in the literatures, the name and the explanation of the defects are sound for the requirements. However, the defects listed in Table II are only possible to be detected and removed from an appropriate document like Software Requirements Specification (SRS). Therefore, in order to handle all defects listed in the table, a complete document must be presented.

Yet, in the process of requirements elicitation in which the requirements are still in high level requirement statements, proper requirement specification document is not there yet. Therefore, this research has scope the list of defects to the followings which is already identified [14] to be feasibility exists in the high level requirements statement during elicitation process. List of the defects are incomprehensible, inconsistent, incomplete, infeasible and incorrect.

III. The Economic Impact of Software Requirements Defects.

Project Managers aim to deliver a product of sufficient quality on time and within budget. In line with that, research has been done to reduce the software requirements defects by detecting and fixing the defects early [15, 16] to better improve overall quality; both in the software development process and the end product. However, Boehm claimed that [17] current software projects spend about 40 to 50 percent of their effort on avoidable rework. Such rework consists of effort spent fixing software difficulties that could have been discovered earlier and fixed less expensively or avoided altogether.

Eliciting and analysing software requirements is a critical stage because defects originated in this stage will propagate to the subsequent stages and therefore effect the entire software development process.

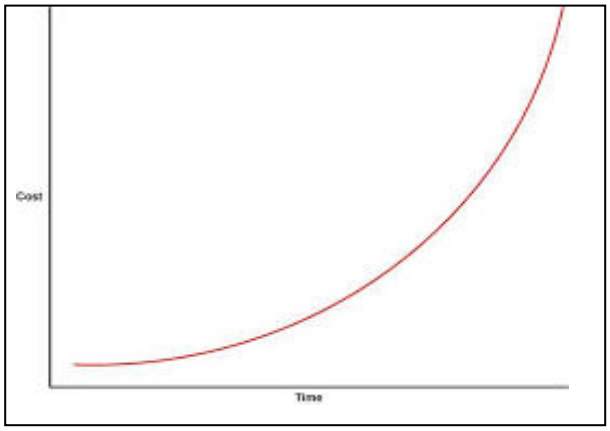

Fig. 1 Cost of fixing Defects [18]

According to Boehm [18] the cost of fixing errors increased exponentially the later the errors were detected in the development lifecycle because the artefacts within a serial process were built on each other. Also, the average cost arose exponentially after the defects were detected because the development continues upon an unstable foundation. This meant that there is a possibility that the software being developed had been based on unwanted or wrong requirements. Fig. 1 shows an exponential effect to the development cost the later the defects were detected and fixed or removed. Recognizing and fixing defects early leads to an economic benefit [16]. It is based on the reduction in future effort of development and to the higher quality inputs on which development and project planning are based. Benefits come in savings of rework when a defect has to be detected and removed at a later stage of development or operation. The benefit from savings depends on the severity of the defect and

the impact it would have had on the development project; this may vary with the development phase in which it would have surfaced [19]. Therefore, it is beneficial for the software development team to being able to recognize defects early.

IV. The Survey Methodology

On top of literature review, a survey has been done to investigate the existence and significance of requirements defects in the industry in Malaysia.

A. Sampling

The sampling frame of this study comprised of various organizations including public sector, private sector and software houses in Malaysia. The targeted population comprised IT practitioners with various experience level which were categorize as fresh graduates (less than a year), junior (1-3 years), senior (3-5 years) and expert (5 years and above). Besides, the IT practitioners have various roles in the software development process which includes business analyst, system analyst, software engineer, tester, programmer and project manager.

The IT practitioners were selected in this study because they are known to undergo the actual activities in software development life cycle. The reason why various roles were involved was due to the diverse practice implemented by different organization and no standard job title or role is pertinent to handle software requirements. For example, in an extreme case, a software engineer could be responsible to elicit the requirements, establish the architecture, design the software, write the code and run the testing himself. Among the 51 respondents, 30 of them claim to be familiar and involved in producing software requirements for the development.

B. The Instrument

The questions were divided into two sections. The first section captured the information of the respondent and the second section captured the defects information.

1) Respondent Information

The respondent information focused on determining the range of respondents by several factors. The first factor is based on different organization types they are working in to represent different practices. The second factor is their years of experience in the industry. The third factor is their role in the software development life cycle and if they are familiar with software requirements.

2) Defects Information

The defects information section focused on determining if the defects really exist from the perspective of the practitioners. If they exist, the types of defects were captured and when the defects usually occurred was identified. Besides, how practitioners handle or manage the defects were recorded too. All items in Defects Information Section, consist of questions 7-10, were measured using four points likert scale from 1=Never to 4=Always.

3) Validity and reliability

In determining the reliability of the instrument, a general rule is that the indicators should have a Cronbach's α of 0.6 or more (Nunnally et al., 1967).With the range of α scores between 0.87 and 0.94 obtained in this study (shown in Table 4), we can conclude that the questionnaire is reliable and the data can be applied for further analysis.

V. The Results and Discussion

This section presents the survey results and discussion.

A. Demographic

Table III shows the demographic profiles of the respondent surveyed. Most of them are from private sector with 41.2% followed by software house (29.4%) and public sector (25.5%). Only 2% are from education organization. The majority of them are junior practitioner with 1 to 3 years of experience (71.8%). Overall, most of the respondents claimed to be familiar with software requirements defects (54.9%) and are involved in producing software requirements documents (58%).

Table III. Demographic profiles of the respondent

	Percentage
Sector	
Education	2.0
Others	2.0
Private	41.2
Public	25.5
Software House	29.4
Involvement in producing Software Requirement	
No (Proceed to Question 4)	42.0
Yes (Proceed to Question 5)	58.0
Level Experience	
Expert (5 years and above)	10.3
Junior (1-3 years)	71.8
Senior (3-5 years)	17.9
Familiar Defect	
No	13.7
No, Not Sure	2.0
Not Sure	29.4
Yes	54.9

B. *Cronbach's α and Statistic*

Table IV presents mean and standard deviation scores of the variables.

Table IV. The results

	Mean	Std. Deviation	N of Items	Cronbach's Alpha
Types of Defects			8	0.871
Missing	2.65	.753		
Incorrect	2.35	.789		
Inconsistent	2.51	.692		
Ambiguous	2.54	.836		
Infeasible	2.19	.877		
Not Relevan	2.16	.688		
Incomprehensible	2.03	.833		
Others	1.92	.829		
Sources of Defects			14	0.938
OperationalDoc	2.97	.164		
SoftwareDoc	2.43	.801		
ReqChangeForm	2.62	.794		
EndUser	2.68	.884		
ManagerialStaff	2.54	.989		
Sponsor	2.03	.866		
CEO/CIO	2.11	.936		
BussinessAnalyst	2.14	.887		
SystemAnalyst	2.32	.784		
EnterpriseArchitect	2.14	.887		
SoftwareDesigner	2.30	.812		
Programmer	2.57	.929		
Tester	2.51	.961		
Others	2.05	1.026		
When the Defects Detected			9	0.883
RequirementPhase	2.45	.961		
ReqElicitation	2.35	.985		
ReqAnalysis	2.48	.851		
ReqSpec	2.58	.923		
ReqValidation	2.71	.824		
DesignPhase	2.77	.762		
ImplementPhase	2.74	.855		
TestingPhase	2.74	.930		
Others	2.16	1.036		

With the range of standard deviation of 0.87 to 0.68, the results show that the respondents agreed on the existence of software requirements defects in the current practice. The types of defects were also agreed to be missing, incorrect, inconsistent, ambiguous, infeasible, not relevant and incomprehensible. Even though there were several feedbacks chose 'others', the defects stated by the respondents were synonyms to the defects listed earlier. Therefore, we can conclude that the types of defects detected by practitioners in the industry are aligned with the justification in the literatures.

Furthermore, the most popular sources of defects were identified as operational documents and requirements change form with mean 2.97 and 2.62 respectively. Other popular sources were identified as stakeholders; end user, managerial staff, programmer and tester with mean more than 2.50.

The results in Table 4 also indicate when the defects are usually detected during the system development lifecycle. In good practice, it is ideal to detect defects as early as possible. However, the results showed that the defects are usually detected later; in design phase, implementation phase and testing phase with mean more than 2.7. Therefore, it is important to instill awareness to encourage defects detection early to avoid unnecessary cost to fix defects later in the system development lifecycle.

C. *Correlation*

Table V shows the correlation between types of defects(ToD), sources of defects (SoD) and when the defects are detected (WDD). Pearson's correlation coefficient *r* is used as a descriptive measurement to test an association between two variables [20]. Despite of correlation coefficients *r* is more than 0.5 with both variables, the sources of defects and when the defects detected has statistically high correlation (r=0.843). Thus, it shows that WDD has a significant association with the SoD. The significant association may suggest that the phase at which the defects were detected is traceable to the sources that cause the defects. All the correlation are significant at P<0.01.

Table V. Correlations

	ToD	SoD	WDD
Types of Defects (ToD)		.514**	.511**
Sources of Defects (SoD)	.514**		.843**
When Defects Detected (WDD)	.511**	.843**	

**. Correlation is significant at the 0.01 level (2-tailed).

In addition, Figure 2 shows scatter plot diagrams of relationship between the three variables based on types of defects (ToD), sources of defects (SoD) and when the defects detected (WDD) respectively. More specifically, these diagrams can visualize potential relationship between two variables individually and show how each point of a variable can respond to another point in another variable. Moreover, the trends of the data scattering also can be identified [21].

Scatter plots for these variables can be done by arranging the variables in pair such as X_i and Y_i. Each pair of Xi and Y_i is plotted in diagram and a regression line can be used for further estimation [22]. Since WDD and SoD has high correlation coefficient as shown in Table 5, the relationship can be observed in Fig.2(b) that most of the data points WDD and SoD are closer to the regression line. Therefore, it would show that the points of when the defects are detected and the sources of defects are closely correlated. Furthermore, distribution of data points between ToD and SoD, and WDD and SOD are likely more scattered about the regression lines as shown in Fig.2(a) and (c). Hence, the relationships both ToD and WDD against SoD are less correlated.

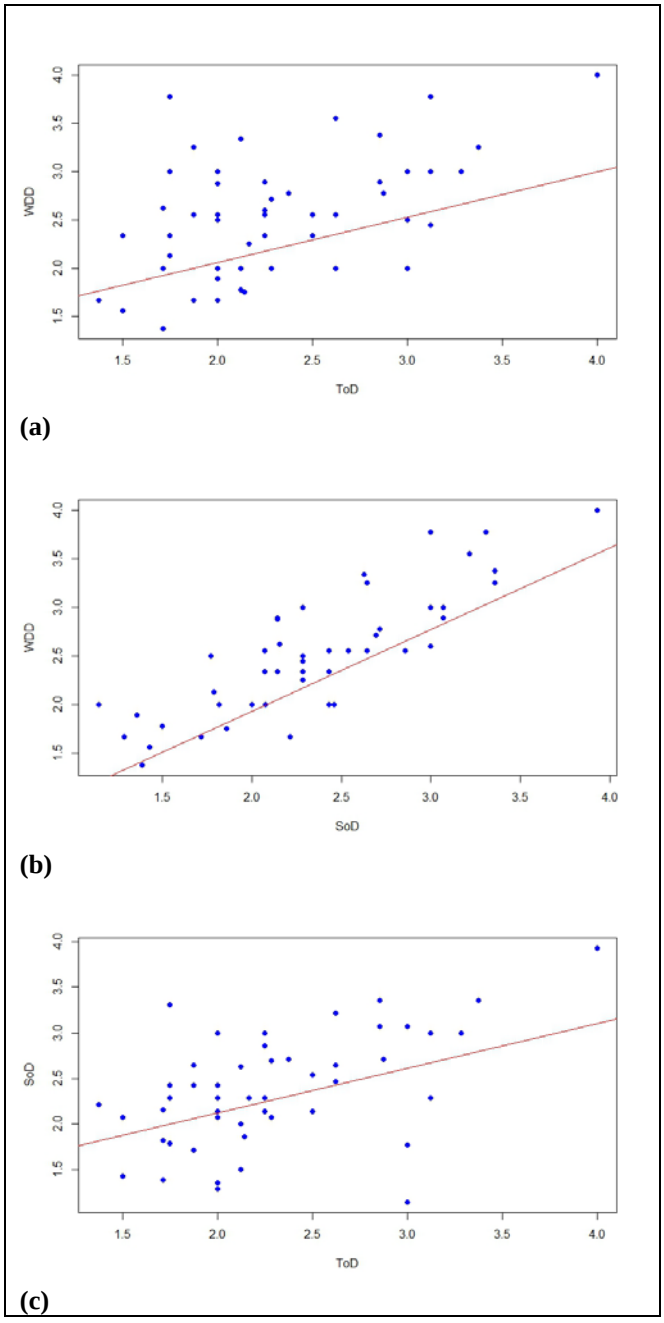

Fig. 2. Scatter plots of three variables (a) type of defect(ToD) against when defects detected(WDD) (b) Sources of Defects (SoD) against when defects detected(WDD) (c) Sources of Defects(SoD) against type of defect(ToD)

Referring to the awareness of IT practitioners to realize the existence of software requirements defects, Fig. 3 shows that there is relatively no difference if they are experienced practitioners or not based on the mean presented. Therefore, even though the majority of the respondents are junior, the data obtain in the survey conducted is reliable.

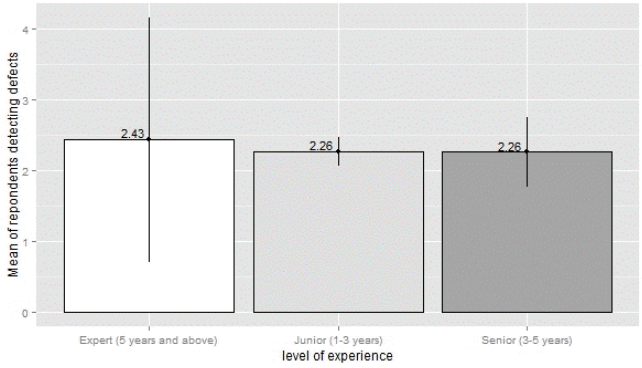

Fig. 3 Mean of respondents detecting defects

VI. Conclusion

Based on the literatures and the survey findings, it is hoped that the knowledge presented in this paper shed some lights to the researchers and academics to realize the significant existence of software requirements defects and the urgency to keep improving the current practice in order to avoid unnecessary cost to fix the defects later. It is also important to realize the significant association between the types of defects and the sources of the defects to serve as a guide for the industry to better handle the defects in the future.

Given the limited sample size and scope within Malaysia, it is reasonable for future study to include a bigger sample size and to open the scope to other countries.

Acknowledgment

The authors would like to acknowledge Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka for contribution to this study.

References

[1] K. Naik, and P. Tripathy, *Software testing and quality assurance: theory and practice*: John Wiley & Sons, 2011.

[2] S. Adikari, C. McDonald, and N. Lynch, "Design science-oriented usability modelling for software requirements," *Human-Computer Interaction. Interaction Design and Usability*, pp. 373-382: Springer Berlin Heidelberg, 2007.

[3] M. V. Mäntylä, and J. Itkonen, "How are software defects found? The role of implicit defect detection, individual responsibility, documents, and knowledge," *Information and Software Technology*, vol. 56, no. 12, pp. 1597-1612, 2014.

[4] D. Pandey, U. Suman, and A. Ramani, "An effective requirement engineering process model for software development and requirements management." pp. 287-291.

[5] J. M. Verner, O. P. Brereton, B. A. Kitchenham *et al.*, "Risks and risk mitigation in global software development: A tertiary study," *Information and Software Technology*, vol. 56, no. 1, pp. 54-78, 2014.

[6] I. L. Margarido, J. P. Faria, R. M. Vidal *et al.*, "Classification of defect types in requirements specifications: Literature review, proposal and assessment." pp. 1-6.

[7] V. Suma, and T. Nair, "Defect Management Strategies in Software Development," *Intec Web Publishers*, 2012.

[8] F. Lanubile, A. Lonigro, and G. Vissagio, "Comparing models for identifying fault-prone software components." pp. 312-319.
[9] G. Walia, and J. Carver, "Development of a Requirement Error Taxonomy as a Quality Improvement Approach: A Systematic Literature Review MSU-070404, Department of Computer Science and Engineering, Mississippi State University," 2007.
[10] IEEE, "Standard Classification for Software Anomalies," 1994, pp. 1044-2009.
[11] S. Kumaresh, and R. Baskaran, "Defect analysis and prevention for software process quality improvement," *International Journal of Computer Applications*, vol. 8, no. 7, 2010.
[12] A. Davis, S. Overmyer, K. Jordan *et al.*, "Identifying and measuring quality in a software requirements specification." pp. 141-152.
[13] N. Yilmatzurk, ""Good Quality" Requirements in Unified Process.," *Engineering and Managing Software Requirements*, A. W. Aurum, C. , ed., Springer-Verlag, 2005, pp. 373-401.
[14] S. Ahmad, "Measuring the Effectiveness of Negotiation in Software Requirements Engineering," The University of Western Australia, 2012.
[15] S. Biffl, B. Freimut, and O. Laitenberger, "Investigating the cost-effectiveness of reinspections in software development." pp. 155-164.
[16] M. Halling, S. Biffl, and P. Grünbacher, "An economic approach for improving requirements negotiation models with inspection," *Requirements Engineering*, vol. 8, no. 4, pp. 236-247, 2003.
[17] B. Boehm, and V. R. Basili, "Software defect reduction top 10 list," *Computer*, vol. 34, no. 1, pp. 135-137, 2005.
[18] B. W. Boehm, *Software engineering economics*: Prentice-hall Englewood Cliffs (NJ), 1981.
[19] S. Biffl, and M. Halling, "Investigating the defect detection effectiveness and cost benefit of nominal inspection teams," *IEEE Transactions on Software Engineering*, vol. 29, no. 5, pp. 385-397, 2003.
[20] R. K. Ekawati, and A. N. Hidayanto, "The Influence of Antecedent Factors of IS/IT Utilization Towards Organizational Performance-A Case Study of IAIN Raden Fatah Palembang," *WSEAS Transactions on Computers*, vol. 10, no. 3, pp. 81-92, 2011.
[21] Z. Nopiahl, M. Khairir, S. Abdullah *et al.*, "Segmentation and scattering of fatigue time series data by kurtosis and root mean square."
[22] R. Mauro, Giuffre, O., Grana, A. & Chiappone, "A Statistical Approach for Calibrating a Microsimulation Model for Freeways," *WSEAS Transactions on Environment and Development*, 2014.

Sabrina Ahmad has qualifications and undergone formal trainings in the area of software engineering and development. She is specialized in requirements engineering in both research and practice. She obtained her PhD in Computer Science from The University of Western Australia in 2012. She endeavours to maintain the bridge between academia and practitioners and therefore continue strengthening software engineering curriculum and apply the knowledge in the industries through consultation projects. Therefore, she continues to upgrade her skills and knowledge through professional training and certification. She obtained Professional Certification in Requirements Engineering and currently cultivate skills in IT Architecture. She is also a Certified Professional IT Architect and formally trained for TOGAF 9.1.

Siti Azirah Asmai received her Ph.D degree in Information dan Communication Technology from the Universiti Teknikal Malaysia Melaka, in 2014. She obtained her master degree in ICT for Engineers from Coventry University, UK in 2004 and bachelor science degree in computer science from Universiti Teknologi Malaysia in 2000. Currently, she is a senior lecturer in the Department of Industrial Computing, Faculty of ICT, Universiti Teknikal Malaysia, Melaka. Her research interests include data analytics, applied statistics, computational industry, simulation and modelling and artificial intelligence.