Journal of Computer Science & Computational Mathematics, Volume 7, Issue 3, September 2017
DOI: 10.20967/jcscm.2017.03.006

A Document-based Traceability Model: An Evaluation by Using Feature Analysis

Azri Azmi*, Suhaimi Ibrahim, Othman Yusop, Haslina Md Sarkan, Kamilia Kamardin, Saiful Adli Ismail

Advanced Informatics School (UTM AIS), Universiti Teknologi Malaysia,
Jalan Sultan Yahya Petra, 54100 Kuala Lumpur
*Corresponding author email: azriazmi@utm.my

***Abstract*:** Software traceability is one of the key quality factors in software development. However, many developers are still not putting enough effort and priority on traceability. In order to verify and validate the completeness and consistency with the requirement, traceability is crucial. This study aims to evaluate the developed traceability model and to compare it with other traceability model with respect to their abilities to support the test management. For this evaluation, the DESMET method will be used, and a feature analysis of the models will be conducted. The evaluation shows that each of the models has some strengths and some weaknesses. Based on the evaluation, it was found that the proposed model meets its expectations to support the test management and its traceability has achieved some promising output and remarkable understanding compared to existing models.

***Keywords*:** software evaluation, software traceability, software documentation.

1. Introduction

Software testing has become more challenging in this era of new technology where product quality is the highest expectation to stakeholders. To develop a high quality of software product, a test manager plays an important role in each of software testing activities [1]. The practice of organizing and controlling the process and documentations essential for the software testing endeavour is called test management. The general objective of test management is to permit development teams to plan, develop, execute and evaluate all testing activities within overall software development life cycle [2]. It is a process of ensuring visibility, traceability and control of testing process to deliver high quality software product [3]. However, very few are working on the importance of test documentation as a way forward to support software testing management.

One of the main challenges in test management is to manage documentations. Test documentation is identified as the pivotal point as stated by the IEEE829:2010 in order to manage and to report the test contents [4]. From another perspective, the test maturity model takes documentation as an important measure to associate the test management to test the process improvement [5]. In other words, one way for test management to remain useful throughout the phases with acceptable maintenance features built-in, is to adopt a good documentation model. Despite this, test documentation is not given due respect by many testers. Test documentation is treated as a time consuming task that not many people would like to get involved with. Because of the lack of staffing, some organizations give less attention on documentation.

In order for understanding the requirement, architecture design, coding, testing and details of intricate applications, software developers rely on documentation to assist them. Lacking such documentation, engineers must depend just on source code. This will result in consuming time and will cause mistakes [6] especially when there are developing systems large scale. Based on Huang and Tiley [7] and Sommerville [8], several shortcomings are observed in recent documentation such as out-of-date and inconsistency between source code and documentation, poor quality and others.

Software traceability is regarded as the key point solution to the above problems. The definition of traceability is the ability of connection between several artefacts in software development phases linking requirements, design, source code and testing artefacts. Today, software traceability has become one of the key attributes to software quality. Unfortunately, many organizations failed to implement effective traceability due to difficulties in creating, assessing, using and maintaining traceability links [9,10]. The accurate traceability practices can help in maintaining software. Thus it will improve the quality of the system as well as the software process. On the other hand, neglecting traceability can lead to reducing the quality of the software product. The quality of the software product cannot be achieved when it is not fully tested and traced with the requirements [11].

In order to verify and to validate the completeness and consistency with the requirements, traceability is crucial. If properly implemented, it can provide significant benefits. Consequently, there must be a careful definition of strategy for implementation of software traceability. There are different organizations or projects for different sets of traceability models, based on business / organizational, domain, project, product or technology [12]. Linkage of data and artefacts, semantic content and automation capability are regarded as the key points of implementing efficacious software traceability. Traceability can be attained when addressing all these aspects. Performing analysis of present model [13], several findings and limitation of existing software traceability models were recognized.

In this paper, a software traceability model evaluation of which aim to help test managers deal with test documents, is presented. It is based on DESMET method. The study utilized feature analysis in evaluating the model to better support test management. Some features needed for processing such models are proposed and every model is scored against each related feature.

This paper is organized into 5 sections. Section 2 describes the evaluation method with feature analysis as an approach used. Section 3 introduces candidate models and set of features utilized and scoring process. Section 4 describes the results obtained from the evaluation made and is followed by conclusion in Chapter 5.

2. Evaluation Method

This section describes the applied evaluation method. A qualitative method based on the evaluation guideline in DESMET is utilized.

2.1 DESMET

DESMET [14] was developed by B.A. Kitchenham in conjunction with the UK Department of Trade and Industry. It is a methodology used to evaluate tools or methods in a domain of software engineering. DESMET was a collaboration project of academia and industry and consists of nine evaluation methods and a set of criteria to help the evaluator choose the most suitable one. An evaluation based on DESMET evaluation is regarded as comparative and context-dependent.

The context identified in this study was choices of methods and tools. Tools, method, combination of tools and method, and project support environment are the objects that can be classified under DESMET methodology. Assuming the DESMET methodology, the first point is to validate the evaluation criteria or technique. The evaluation criteria can be categorized according to the aspects of the tool that we are going to assess. In the DESMET methodology, there are qualitative and quantitative evaluation forms. Both sets of evaluation methods may be prepared as a case study, formal experiment, or survey. Qualitative case study so called Feature Analysis is chosen for the purpose of qualitative evaluation in this project.

2.2 Feature Analysis

Feature analysis is a qualitative form of evaluation that is performed by the person who used a tool or method on a real project so called feature analysis case study. It is an established evaluation method in software engineering field [14–16]. It is scaled with the degree of support that a tool provides with a set of features. The objective of this evaluation is basically to provide an input into decision about purchasing a method or tool to be used by an organization. Therefore the significant part of this evaluation is to: i) help in identifying the important features of the method/tool, ii) help in identifying the differences between methods and tools, and iii) providing an explanation of why a decision was made. Based on DESMET method and works by Hedberg [17], there are three steps in feature analysis: i) feature identification, ii) feature scoring/ranking and iii) analyzing and interpreting the results. Since DESMET method provides only general set of evaluation criteria, we need to identify significant criteria from other researchers. Table 1 illustrates the software evaluation features identified from the features mapping process.

Table 1. Selection of Features for Software Development Tool Evaluation

Features	Sub-Features	Description	Author/Reference
Traceability	• Tracing Type • Tracing Phase	The ability of the product to be traced forward and backward	Shahid [18] Meneely *et al.* [19] Kornecki & Zalewski [20] Illes *et al.*[21]
Document/Artefact Management	• Requirement • Design • Source Code • Test Suite • Others	The ability of the product to integrate all software engineering documentation and to check the completeness and consistency between them	Hedberg & Lappalainen [17] Grimán *et al.*[16]
Defect management	• Defect list • Defect tracking	The ability of the product to locate defect and make a report	Hedberg & Lappalainen [17]
Report	• Report generation	The ability of the product to produce a report	Shahid [18] Marshall *et al.* [15] Marshall *et al.* [22]
Activity support	• Data extraction • Test Management	The ability of product to support any activity that is suggested	Marshall *et al.* [15] Marshall *et al.* [22]

3. Traceability Models, Scoring and Features

In this section there is the overview of software traceability models, features rating, level of importance, feature set and overall scores.

3.1 Traceability Models

There are six traceability models utilized in this study:

a) **Inter Requirements Traceability Model (IRT**) which trace requirements-to-requirements and support for change impact analysis [23].
b) **Total Traceability Model (TT)** that uses horizontal and vertical traceability in order to form a total traceability between artefacts such requirements, test cases, design and codes. TT supports top down and bottom up traceability from component perspectives [24].
c) **Coding Phase Requirements Traceability Model (CPRT)** which traces between requirements and source code only [25].
d) **End-to-end Traceability Model (ETET)** which traces artefacts in the whole SDLC. It only supports post-requirements traceability and not in the maintenance phase [26].
e) **Traceability Web Model (TW)** which traces artefacts at different level of granularity [12].
f) **Document-Based Traceability Model (DBT)** (the proposed model) which uses document-based link to trace link between software testing artefact and requirements. DBT is aimed to support test management.

3.2 Features Rating

This sub-section will describe the elements of the scoring process as: i) to score features in order to produce a raw score, ii) to assign a level of importance for every feature, and iii) to calculate scores for each features. Basically there at two types of features rating: simple features and compound features. Simple features are assessed with YES/NO answer. Wherever a future is presented or powerfully supported, it will be awarded with a score of 1, and if it is partially presented or partly supported, the score is 0.5. Otherwise, if the future is absent, it will be set as 0. Table 2 illustrates the marking scale.

Table 2. Scoring scale

Is the feature present?	Score
Yes	1
Partially	0.5
No	0

Meanwhile, compound features are measured on an ordinal scale based on the degree of support offered by the method or tool. Every simple feature can be accompanied by a level of important assessment while every compound feature can be accompanied by a level of importance and conformance with a suitable scale based on particular feature or characteristic. Level of importance will be discussed next.

3.2.1 Importance Level

A method or tool is effective when it incorporates the most vital feature to its clients. The importance of a feature can be evaluated by considering whether it is mandatory or just desirable. If a tool neglects incorporating a mandatory feature, then it is by definition unacceptable. Non-mandatory features are characterized as desirable. This perspective of importance prompts two evaluation criteria: one of which distinguishes the necessity of a component regardless of whether it is mandatory; the other surveys the degree to which a non-mandatory feature is sought. For this study, a feature is categorized as in Table 3 with the multiplier (for example the weighting) related to each level of importance. Meanwhile, Table 4 shows the set of features with level of importance.

3.2.2 Overall Scores and Feature Set

As specified before, the score for every feature is determined through duplicating the score by the importance weighting for that feature. To recognize a percentage score for each list of features (as appeared for instance in column 5 of Table 6), the weighted scores can be combined. The feature set rate score is calculated in the following equation:

$$\% \; score = \frac{\sum of \; Weighted \; Scores}{Maximum \; Score} \quad x \; 100 \qquad (1)$$

Table 3. Importance Level of a Feature with Multiplier

Importance	Abbreviation	Multiplier
Mandatory	M	*4
Highly Desirable	HD	*3
Desirable	D	*2
Nice to have	N	*1

If all features in the set are fully presented or supported the feature will get a maximum score. For instance, Feature Set 3 (F3) includes two sub-features (F3-SF01 and F3-SF02). F3-SF01has been labeled as Mandatory (M). This means the maximum score of this sub-feature is 4. In the meantime, F3-SF02 has been categorised as Highly Desirable (HD) and gets 3 for the score. Consequently, the maximum scores for F3 are 7.

Likewise, the maximum scores of the residual feature sets are 15, 7, 2 and 5 for Feature Sets F2, F3, F4, and F5 respectively. Through taking an average (weighted) of the rate scores for each feature set, an overall percentage score for every model can be stated. Normalized score is used because there are different numbers of sub-features in the feature set.

Table 4. Features and Sub-Features utilized in the analysis

Id	Feature Set		Id	Sub-feature	Sub-feature Level of Importance
F1	Traceability	Type	F1-SF01	Manual	M
			F1-SF02	Semi	HD
			F1-SF03	Automation	D
		Phase	F1-SF04	Development	M
			F1-SF05	Maintenance	HD
F2	Document Management		F2-SF01	Requirement	M
			F2-SF02	Design	HD
			F2-SF03	Source Code	HD
			F2-SF04	Test Suits	M
			F2-SF05	Others	N
F3	Defect Management		F3-SF01	Defect list	M
			F3-SF02	Defect tracking	HD
F4	Report		F4-SF01	Report Generation	D
F5	Activity Support		F5-SF01	Data extraction	D
			F5-SF02	Test Management	HD

Table 5 shows the feature set weighting used in this research. The values give emphasis to feature set F1 and feature set F2. The general score for each model can be determined using the following equation:

$$Overall\ Score = \frac{\Sigma_{i=1}^{5}(w_i TP_i)}{\Sigma_{i=1}^{5}(w_i)} \quad (2)$$

where,

w_i is assumed as the weighting for the i[th] feature set

TP_i will be the % score for the i[th] feature

Table 5. Feature Set Weighting

Feature Set	Weight
F1	0.4
F2	0.3
F3	0.2
F4	0.1
F5	0.1

4. Results

The overall scores for each of the model is reported in this section. The outcomes of feature analysis for the entire model are condensed in Table 6. This study is intended to evaluate the capability of model from the viewpoint of test management. Therefore, we picked the overall weights values which underscored the capabilities that give the capacities required by the author. Next, we will discuss the features score in detail. (Only Feature Set 1 will be discussed here)

4.1 Feature Set 1 Scores

Since traceability is the crucial part in evaluating all the models, therefore the total weighted score for this feature is 16 and consists of 5 sub-features: F1-SF01-F1-SF05. This sub-features help in analyzing, managing, controlling and applying changes in software traceability. The mechanisms for this objective may be manual, semi-automated or automated. The sub-features also identified phases in which the traceability needs to be created or updated: development and maintenance. Three models (TT, ETET and TW) scored 15 out of 16 with percentage of 93% by using equation1. Most of the models are not capable to generate traceability link automatically. Since automation sub-feature level of importance is D with full marks of 2, none of the models were achieved it. The sub-features F1-SF01 and F1-SF02 are characterized as manual and semi-automatic traceability types. All the models scored full marks in these categories. IRT and CPRT model scored 11 out of 16 since the models cater traceability for requirements to requirements only. The DBT gained 14 marks since it is not capable to generate links automatically. Mostly, all the models are implemented in the development phase accept for IRT and CPRT are not capable in the maintenance phase. Figure 1 represents graph that tabulates the score of set feature 1. Meanwhile, Figure 2 shows the percentage of feature set score.

4.2 Overall Scores

This section discusses the outcome of the feature analysis. Table 6 summarizes the overall scores and the feature set scores. As depicted in Table 6, DBT attains the highest overall score of 84.4% with score of 37.5 out of 45, thus

can be considered as the most suitable model to support the test management. CPRT has the lowest overall score approximately 44.3% with the total score 26 out of 45. This is due to the characteristics of CPRT which traces for only requirements to requirements, and loose marks in document management feature set. CPRT scores 4 out of 15. The second lowest is IRT with a total score of 20.5 out of 45. With the usage of feature set weightings, the score is about 52.3% for overall percentage score. In this case, the IRT traces between requirements and source codes and not with other documentations. The next model, ETET with overall percentage score of 62.2% and the total score for this model is 26 out of 45. With partial scores for F2-SF02, F2-SF03 and F2-SF04, this gives an advantage to ETET.

Table 6. The Overall Scores Details of Feature Analysis

Traceability Model		IRT			TT			CPRT			ETET			TW			DBT		
Feature Set	Sub Feature	WS	FSS	% FSS	WS	FSS	% FSS	WS	FSS	% FSS	WS	FSS	% FSS	WS	FSS	% FSS	WS	FSS	% FSS
F1	F1-SF01	4	11/16	68	4	15/16	93	4	11/16	68	4	15/16	93	4	15/16	93	4	14/16	87
	F1-SF02	3			3			3			3			3			3		
	F1-SF03	0			1			0			1			1			0		
	F1-SF04	4			4			4			4			4			4		
	F1-SF05	0			3			0			3			3			3		
F2	F2-SF01	4	5.5/15	37	4	12/15	80	4	4/15	27	4	9/15	60	4	10.5/15	70	4	13.5/15	90
	F2-SF02	0			3			0			1.5			3			1.5		
	F2-SF03	1.5			3			0			1.5			1.5			3		
	F2-SF04	0			2			0			2			2			4		
	F2-SF05	0			0			0			0			0			1		
F3	F3-SF01	0	0/7	0	0	0/7	0	0	0/7	0	0	0/7	0	2	2/7	28	4	5.5/7	78
	F3-SF02	0			0			0			0			0			1.5		
F4	F4-SF01	2	2/2	100	2	2/2	100	1	1/2	50	1	1/2	50	2	2/2	100	2	2/2	100
F5	F5-SF01	2	2/5	40	1	1/5	20	1	1/5	20	1	1/5	20	2	2/5	40	1	2.5/5	50
	F5-SF02	0			0			0			0			0			1.5		
Total Score		20.5/45			30/45			26/45			26/45			31.5/45			37.5/45		
Overall % Score Using Feature Set Weightings		52.3%			73.2%			44.3%			62.2%			69.80			84.4		

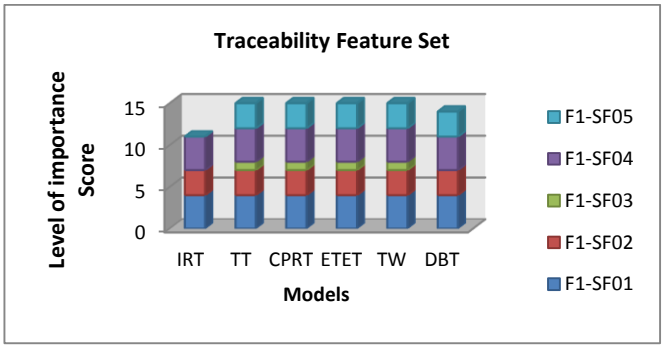

Figure 1. Multiple Metric Graph for Feature Set Score

The following model is TW. With partially score for F3-SF01 which is defect list sub-feature, total score for TW is 31.5 out of 45 with percentage of 69.80%. TT can be a competitive to DBT with score of 73.2%. TT score 13 out of 45 and loose mark in feature set 3 where the model do not support defect management. A variation of Kiviat diagram called Multi Metric graph is used to illustrate the feature set scores and overall scores. Figure 3 illustrates the feature set score using multi metric graph while Figure 4 illustrates the overall percentage score using feature set weightings. Meanwhile, Figure 5 shows the line chart to illustrate percentage feature score versus model.

5. Conclusion

This study has evaluated six traceability models using the DESMET method. A set of features which such models would possess, has been settled and used as a criterion against those to evaluate the models. The purpose of the evaluation is to justify if the models can achieve its effectiveness and accuracy dealing with document-based traceability. The scored results from feature analysis show that the proposed model gets the highest score compared to others. The results show that each of the models has some strengths and some weaknesses. Based on the evaluation, it was found that the proposed model meets its expectations to support the test management and its traceability has achieved promising output and remarkable understanding as compared to other models.

Acknowledgements

The study is financially supported by Research University Grant Tier 1 (Vot Number 11H24), Universiti Teknologi Malaysia and Ministry of Higher Education Malaysia (MOHE).

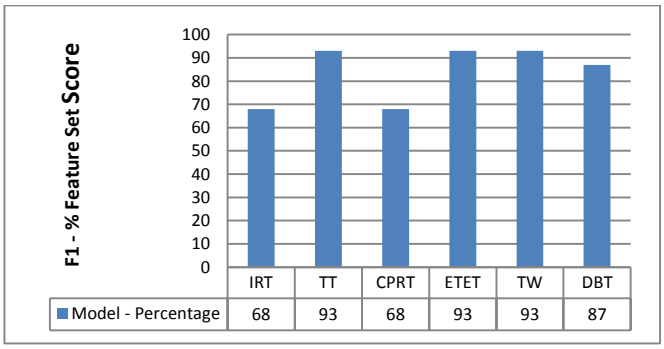

Figure 2. F1 Percentage of Feature Set Score

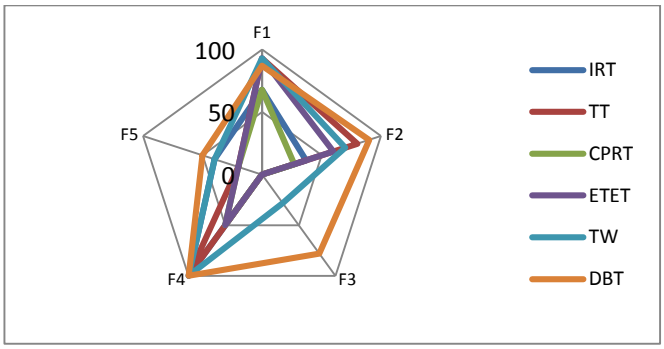

Figure 3. Multiple Metric Graph for Feature Set Score

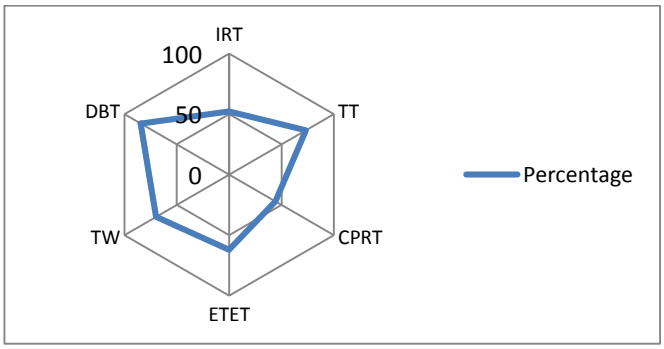

Figure 4. Multiple Metric Graph for Overall % Score Using Feature Set Weightings

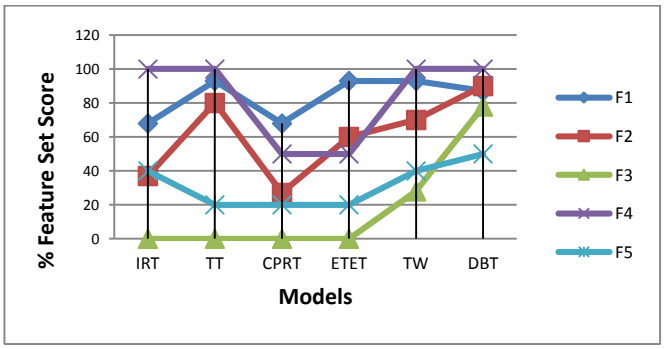

Figure 5. % Feature Set Score Versus Models

References

[1] S. Eldh, J. Brandt, M. Street, H. Hansson, S. Punnekkat, "Towards Fully Automated Test Management for Large Complex Systems", in *2010 Third International Conference on Software Testing, Verification and Validation*, 2010, pp. 412–420.

[2] L. Gao, "Research on implementation of software test management", *2011 3rd International Conference on Computer Research and Development (ICCRD)*, 2011, vol. 3, pp. 234–237.

[3] A. Lodhi, S. Wind, K. Turowski, "Test Management Framework for Managing IT Projects in Industry", *2013 IEEE 10th International Conference on e-Business Engineering (ICEBE)*, 2013, pp. 509–514.

[4] R. M. Sidek, A. Noraziah, M. H. A. Wahab, "The Preferable Test Documentation Using IEEE 829", *Software Engineering and Computer Systems*, Springer, 2011, pp. 109–118.

[5] E. Van Veenendaal, J. J. Cannegieter, "Test Maturity Model integration (TMMi)", *TMMi Foundation*, 2010.

[6] B. Thomas, S. Tilley, "Documentation for software engineers: what is needed to aid system understanding?", *Proceedings of the 19th annual international conference on Computer documentation*, 2001, p. 236.

[7] S. Huang, S. Tilley, "Towards a documentation maturity model", *Proceedings of the 21st annual international conference on Documentation*, San Francisco, CA, USA, 2003, pp. 93–99.

[8] I. Sommerville, *Software engineering, vol 2: the supporting process*. Addison-Wesley, ISBN 0-321-31379-8, 2002.

[9] A. Knethen, B. Paech, "A survey on tracing approaches in practice and research", *IESE-Report No. 095.01/E*, vol. 95, Jan. 2002.

[10] B. Ramesh, M. Jarke, "Toward reference models for requirements traceability", *IEEE Transactions on Software Engineering*, vol. 27, no. 1, pp. 58–93, 2001.

[11] O. M. Yusop, S. Ibrahim, "Software Maintenance Testing Approaches to Support Test Case Changes–A Review", *Digital Information and Communication Technology and Its Applications*, Springer, 2011, pp. 33–42.

[12] V. Kirova, N. Kirby, D. Kothari, G. Childress, "Effective requirements traceability: Models, tools, and practices", *Bell Labs Technical Journal*, vol. 12, no. 4, pp. 143–158, 2008.

[13] A. Azmi, S. Ibrahim, "Implementing Test Management Traceability Model to Support Test Documents", *International Journal of Digital Information and Wireless Communications (IJDIWC)*, vol. 1, no. 1, pp. 109–125, 2011.

[14] B. Kitchenham, S. Linkman, D. Law, "DESMET: a methodology for evaluating software engineering methods and tools", *Computing & Control Engineering Journal*, vol. 8, no. 3, pp. 120–126, 1997.

[15] C. Marshall, P. Brereton, B. Kitchenham, "Tools to support systematic reviews in software engineering: A feature analysis", *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, 2014, p. 13.

[16] A. Grimán, M. Pérez, L. Mendoza, F. Losavio, "Feature analysis for architectural evaluation methods", *Journal*

of Systems and Software, vol. 79, no. 6, pp. 871–888, 2006.

[17] H. Hedberg, J. Lappalainen, "A preliminary evaluation of software inspection tools, with the DESMET method", *Quality Software, 2005.(QSIC 2005). Fifth International Conference on*, 2005, pp. 45–52.

[18] M. Shahid, *A Traceability Approach for Hybrid Coverage Analysis to Support Software Maintenance*, University Teknologi Malaysia, Kuala Lumpur, 2014.

[19] A. Meneely, B. Smith, L. Williams, "Validating Software Metrics: A Spectrum of Philosophies", *ACM Trans. Softw. Eng. Methodol.*, vol. 21, no. 4, pp. 24:1–24:28, 2013.

[20] A. J. Kornecki, J. Zalewski, "Experimental evaluation of software development tools for safety-critical real-time systems", *Innovations in Systems and Software Engineering*, vol. 1, no. 2, pp. 176–188, 2005.

[21] T. Illes, A. Herrmann, B. Paech, J. Rückert, "Criteria for software testing tool evaluation. a task oriented view", *Proceedings of the 3rd World Congress for Software Quality*, 2005, vol. 2, pp. 213–222.

[22] C. Marshall, P. Brereton, B. Kitchenham, "Tools to support systematic reviews in software engineering: a cross-domain survey using semi-structured interviews", *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, 2015, p. 26.

[23] M. Narmanli, *A business rule approach to requirements traceability*, Middle East Technical University, Ankara, Turkey, 2010.

[24] S. Ibrahim, N. B. Idris, M. M. UK, A. Deraman, "Implementing a document-based requirements traceability: A case study", *IASTED International Conference on Software Engineering*, 2005, pp. 124–131.

[25] A. M. Salem, "Improving software Quality through requirements traceability models", *Computer Systems and Applications, 2006. IEEE International Conference on.*, 2006, pp. 1159–1162.

[26] H. U. Asuncion, F. François, R. N. Taylor, "An end-to-end industrial software traceability tool", *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, Dubrovnik, Croatia, 2007, pp. 115–124.