Proceedings of the International Multiconference on
Computer Science and Information Technology pp. 53–58
ISBN 978-83-60810-22-4
ISSN 1896-7094

Selecting the best strategy in a software certification process

V. Babiy, R. Janicki, A. Wassyng
Department of Computing and Software
McMaster University
Hamilton, ON., Canada

A. D. Bogobowicz
Faculty of Mathematics
and Natural Sciences
University of Cardinal
Stefan Wyszynski
Warsaw, Poland

W. W. Koczkodaj
Department of Mathematics
and Computer Science
Laurentian University
Sudbury, ON., Canada

***Abstract*—In this paper, we propose the use of the pairwise comparisons (PC) method for selection of strategies for software certification. This method can also be used to rank alternative software certification strategies. The inconsistency analysis, provided by the PC method, improves the accuracy of the decision making. Some current methods of software certification are presented as they could be modified by the proposed method. Areas of potential future research are discussed in order to make the software certification process more feasible and acceptable to industry.**

I. Introduction

THE PROCESS of software certification is time consuming and hence expensive. Most software systems, since they are usually not critical, never go through a certification process. Thus software is still being developed without any consideration given to software certification [1]. In most modern systems, software is one of the most complex components. At the same time it is considered as one of the most error prone, despite the increasing demand for reliable software. As a result, there is an apparent need for a viable dynamic software certification process which can adjusted to the dynamic demands of the rapidly evolving software industry [2]. We propose to use the pairwise comparisons method as a means of selecting a strategy for a software certification process. It is quite likely that a certifying body would need to adjust their certification strategy for almost every project. This is because projects are different, especially when they are designed for different domains. A better software certification strategy may require a smaller amount of resources in order to adjust to new scenarios. The use of this approach will provide insight and understanding of the software certification process. For every project, some properties will be more important than others, while some properties will be completely irrelevant. The pairwise comparisons method can provide a scheme where the software certification strategy can be modified easily and adapted to different scenarios.

II. Product Based Certification

The objective of product based certification is to deduce whether the product conforms to requirements and provide an evaluation of the developers abilities to produce new products while conforming to requirements [3]. ISO IEC 14598 provides instructions on how to evaluate a software product. It uses the ISO IEC 9126 standard which describes how general attributes can be subdivided into less general attributes. In practice, both standards are applied in parallel. ISO IEC 14598 has four phases: defining the evaluation requirements, identifying the evaluation, building the evaluation schedule and executing the evaluation schedule. In defining the evaluation requirements, attributes and sub-attributes for the product evaluation are defined. These attributes and sub-attributes could be taken from the McCall's and Blundell's quality models [4], [5]. In identifying the evaluation, a collection of metrics are defined for the evaluation of all attributes and sub-attributes. In addition, metrics which will evaluate relationships between a product and its environment are also defined. While building the evaluation schedule, a detailed evaluation plan is constructed. Finally, the evaluation schedule is executed [6].

III. Process Based Certification

The IEC 61508 (Functional safety of electrical/electronic/programmable electronic safety-related systems) and DO-178B (Software Considerations in Airborne Systems and Equipment Certification) standards follow a process based methodology and thus implicitly view the software certification process as one that places the emphasis on checking that an approved process was adhered to. These standards describe the collection of practices which should be followed during software development. They claim that it would be easier to achieve validation and verification of software by following the proposed practices. The IEC 61508 and DO-178B standards should be followed in correlation with other regulations where they outline the significance of software failure. The Development Assurance Levels (DALs), from the domain of civil aerospace, are an example of this correlation which dictate levels of criticality. The automotive and European rail industries use Safety Integrity Levels (SILs). The (DALs) and (SILs) are not similar in their applications, despite the strong tendency for them both to focus on risk reduction. The more critical the software, the greater the need for risk reduction to be an essential attribute of the software. Greater demands upon the (SILs) and (DALs)

lead to stricter demands from the software development process. The DO-178B argues that the verification of a system should be accomplished through extensive testing, while highly emphasizing the need for a good traceability process and a manual review of the components [7]. The verification process supported by DO-178B is subdivided into four levels. There are twenty eight properties at the lowest level, D. They validate tools, high level requirements, and the configuration of the development process. The next level, C, deals with twenty nine properties. They validate low level requirements, testing and code coverage. The next level, B, deals only with eight properties and logic. The highest level, A, is responsible for sixty six attributes. At this level, while the overall quality of the product is evaluated, a significant focus is allocated to traceability [7].

IV. Motivation

In situations where it is difficult or infeasible to use an algorithm, we revert to the use of heuristics in order to find solutions. There are a large number of attributes which should be considered during the certification process. As projects evolve rapidly and grow in complexity we need mechanisms to assign consistent weights to attributes and properties. The pairwise comparisons method is ideal for this task because it can reduce inconsistencies while still maintaining some acceptable margin of error. We describe a process on how to assign consistent weights to attributes and properties. Once inconsistency is minimal, preferably not zero, the developed software certification strategy can be used as a dynamic entity in the software certification process [8], [9].

V. Pairwise Comparisons Method

The pairwise comparisons method was used for the first time in 1785 by Condorcet. He used this method in the election process where voters rank candidates based on their preference [10]. The method was a voting system which used matrices for particular pairwise comparisons with rows representing each candidate as a runner and columns representing each candidate as an opponent. It was Fechner who specified pairwise comparisons as a scientific method in 1860, although only from the psychometric perspective [11]. Thurstone, in 1927, provided a mathematical analysis of this method and called it the *law of comparative judgments* [12]. The *law of comparative judgments* can be used to scale a collection of attributes based on simple comparisons between attributes taken two at a time. Although, Thurstone referred to it as a law, it can be more appropriately identified as a measurement model which could be of important use for software certification. This model allows experts to synthesize diverse procedures involved in software certification. The hierarchy reduces the number of comparisons from $O(n^2)$ to approximately $O(n \ln n)$, making it applicable to a wide variety of problems. For example, a moderate case with 49 features would require 1,176 comparisons without a hierarchy and only 168 comparisons if these 49 features are arranged into a hierarchy by grouping seven features. Measurements of length, such as a meter or foot, or by mass and weight are commonly used and accepted. Society has become accustomed to having standards for the majority of tasks, and sometimes it is difficult to understand standards, which often occur in the software industry, without an acceptable universal measuring method. In the case of software certification, many strategies may need to be developed for a single project. It is safe to conclude that developing a single certification strategy is not feasible and would not work for all types of projects, because some projects have very little in common [9], [13], [14].

TABLE I: Comparison scale

Code	Definition of intensity or importance
1	Equal or unknown importance
2	Weak importance of one over another
3	Moderate to essential importance
4	Demonstrated importance
5	Absolute importance
3.5 etc	Intermediate importance

A. Inconsistency Analysis

The pairwise comparisons method does not impose any limit on the number of criteria. Setting the maximum number of entities on one level to seven is accepted as a heuristic, because seven items gives 21 distinct pairs to compare. The first step in pairwise comparisons is to establish the relative preference of two criteria for situations in which it is impractical or irrelevant to provide absolute estimations. The relative comparison coefficients a_{ij} for criteria $C_1, C_2, \ldots, C_n$ are expected to satisfy $a_{ii} = 1$ and $a_{ij} = 1/a_{ji}$. The first constraint is related to comparing a given attribute with itself. The second constraint is a consequence of the obvious fact that $x/y = 1/(y/x)$ for $x, y \neq 0$. A scale from 1 to 5, as demonstrated in Table I, is used for expressing the importance of one attribute over others. This is accomplished in a pair. Other scales also exists, but as described by [8] larger values lose meaning in the comparison process.

The absolute estimation of the weights defining the importance of analyzed software certification criteria is practically unobtainable through either statistical or formal procedures. It would be beneficial to have experiments which may contribute to the accuracy of the estimates. However, it is unrealistic to expect such experiments to take place. This approach allows us to improve the processing of often subjective expert assessments in the certification process. We propose the use of a comparison scale that is demonstrated in Table I for the subjective expression of relative preference.

Reference	Criterion
C_1	Functionality
C_2	Reliability
C_3	Usability
C_4	Efficiency
C_5	Maintainability
C_6	Portability

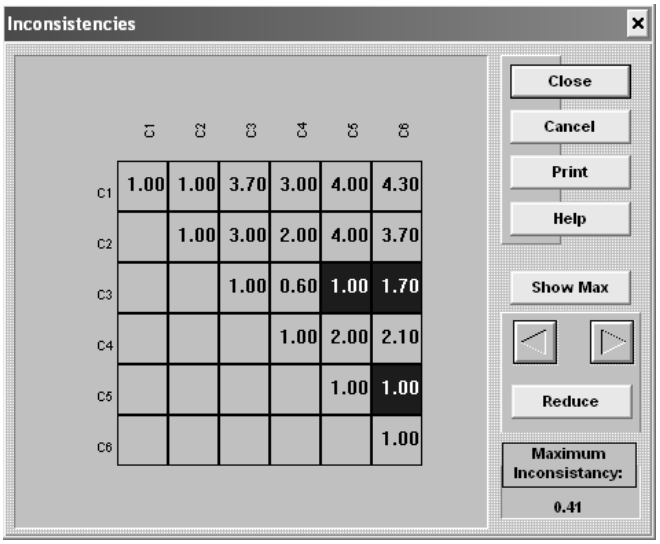

Fig. 1: Relative importance of considered software quality attributes

The values of relative importance, which are given in Figure 1, have been entered by a single person solely for demonstration of the method. We have used the concluder software which can be download from the website that is maintained by W. W. Koczkodaj. [1] The values were deduced from the comparison in pairs. In a real scenario, the values should be reasoned about by a team of experts. The attributes have been taken from the ISO/IEC 9126 software standard. They are also known as the top six level attributes, which are considered to be key attributes for software quality [15]. Now we consider the process to identify the best alternative. Let us denote the attributes by $A_1, A_2, \ldots, A_n$ (n is the number of compared attributes), their actual weights by $\gamma_1, \gamma_2, \ldots, \gamma_n$, and the matrix of the ratios of all weights by $\Gamma = [\gamma_i/\gamma_j]$. The matrix of pairwise comparisons $\mathrm{M} = [a_{ij}]$ represents the assessments between individual pairs of alternatives ($\mathrm{M_i}$ versus $\mathrm{M_j}$, for all $i, j = 1, 2, ...n$) chosen usually from a given scale. The elements a_{ij} are considered to be estimates of the ratios γ_i/γ_j, where γ is the vector of actual weights of the attributes. All the ratios are positive and satisfy the reciprocity property $a_{ij} = 1/a_{ji}$, $i, j = 1, 2, \ldots, n$. The inconsistency concept was explained in [16]. The distance based inconsistency indicator is defined as the maximum over all triads $\{a_{ik}, a_{kj}, a_{ij}\}$ of elements of M (with all indices i, j, k distinct) of their inconsistency indicators. It is defined as:

$$ii = \min\left(\left|1 - \frac{a_{ij}}{a_{ik}a_{kj}}\right|, \left|1 - \frac{a_{ik}a_{kj}}{a_{ij}}\right|\right) \quad (1)$$

Three is the minimal number of attributes which may cause inconsistency. Comparing two attributes will often lead to inaccuracy. The distance based inconsistency is the minimum distance from three ideal triads with no inconsistency when the third value is substituted using the consistency condition $a_{ij} \times a_{jk} = a_{ik}$. Since we are not in a position to determine which ratio is incorrect, all three assessments must be reconsidered before we attempt finding a consistent approximation for a given pairwise comparisons matrix. The stress on localizing the most inconsistent assessments is expressed by adding the *consistency-driven* to the name of the method since it is easier to remedy implications of an error when we are able to localize it. There is no practical reason to continue decreasing the inconsistency indicator to zero. Only the high values of the inconsistency indicator are considered as unacceptable and harmful. A very small value, or zero, may indicate a faked result rather than a true estimate. The practical challenge in working with the pairwise comparisons method comes from the lack of consistency of the pairwise comparisons matrices. Depending on the strategy it may take some time to get the matrix consistent [9], [13], [14], [17].

VI. Demonstration of the Matrix Adjustment

Assume the following attributes are considered for evaluation: safety, security, reliability, resilience, robustness, knowing, testability, adaptability, modularity, complexity, portability, usability, reusability, efficiency and learn-ability. They are considered in [18] as a general group of attributes of any software. All the attributes are subdivided into two main categories, such as development and maintenance. These groups are subdivided further and weights are assigned as demonstrated in Table II. It is safe to assume that some areas of software evolution are based on intuition and experience. In situations where there is more than just one person making decisions there is a greater possibility for inconsistency to occur. Industry must rely on the subjective judgments of experts in situations where practical methods of measure are unknown [9], [13].

TABLE II: On the left inconsistent strategy and on the right consistent strategy

Attribute	Percent	Attribute	Percent
development	**80%**	**development**	**80%**
efficiency	17.40%	efficiency	17.40%
resilience	8.66%	resilience	8.66%
robustness	4.96%	robustness	4.96%
adaptability	3.78%	adaptability	3.78%
modularity	46.23%	modularity	46.23%
complexity	**26.58%**	**complexity**	**19.19%**
portability	**14.06%**	**portability**	**17.29%**
reusability	**5.58%**	**reusability**	**9.74%**
reliability	16.37%	reliability	16.37%
knowing	9.14%	knowing	9.14%
safety	5.23%	safety	5.23%
security	2%	security	2%
maintenance	**20%**	**maintenance**	**20%**
usability	11.49%	usability	11.49%
learn-ability	4.95%	learn-ability	4.95%
testability	3.56%	testability	3.56%

[1] website to download concluder: http://www.cs. laurentian.ca/wkoczkodaj/

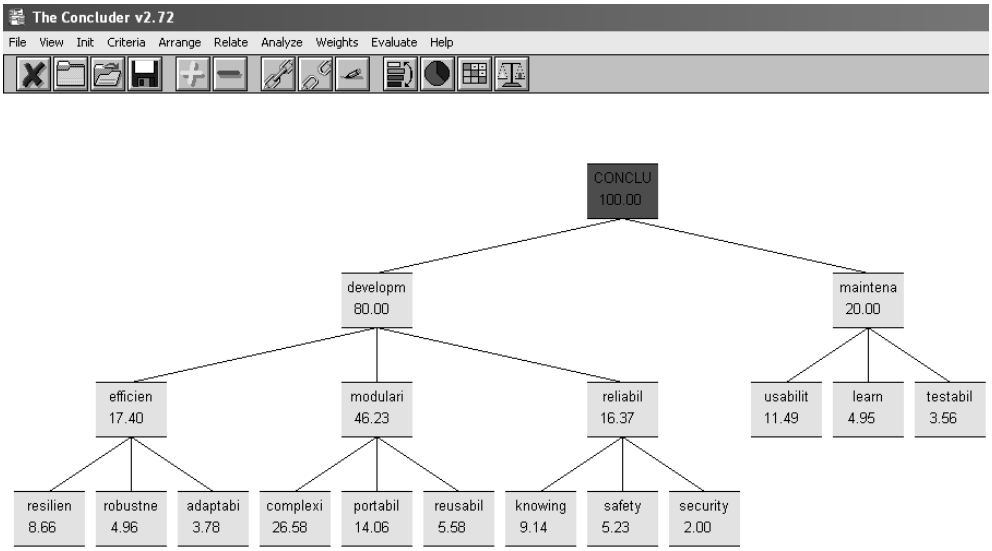

Fig. 2: Inconsistent strategy.

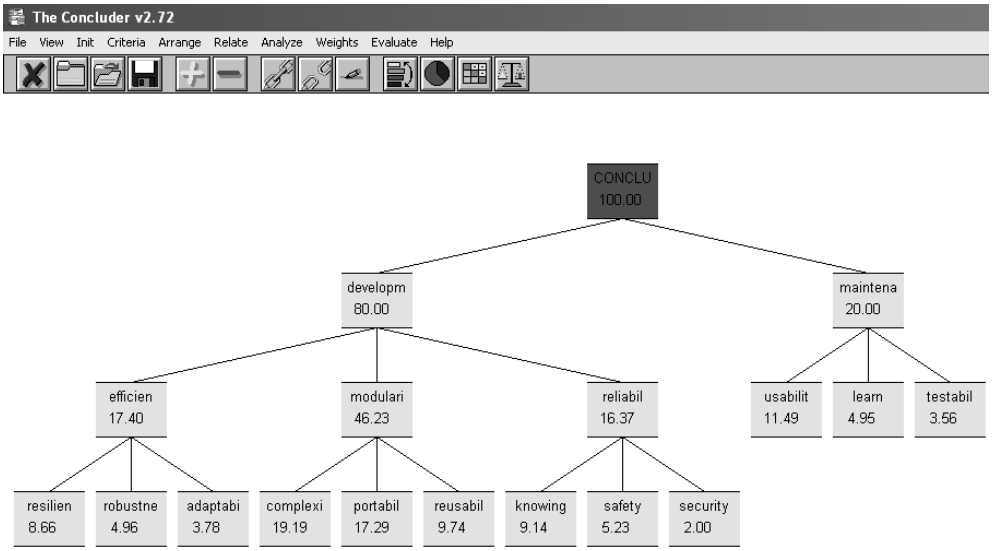

Fig. 3: Consistent strategy.

From Table II we can evaluate the complexity, portability and reusability triad where C_1 = complexity, C_2 = portability and C_3 = reusability. This triad has an inconsistency of 0.75 which is shown in Figure 4. As described in [8] it is not recommended. According to [8] the acceptable inconsistency is around 0.33. We have to adjust the values in order to bring the inconsistency down. After the adjustment, and as demonstrated in Figure 5, the inconsistency has decreased to 0.3 which is more acceptable.

In Table II, the weights of the attributes are allocated based on the significance of each attribute, and the most important criteria is complexity. After the correction we can see a new percentage redistribution, which is shown in Table II. The appropriate concluder's strategy models which demonstrate the percentage redistribution are given in Figure 2 and Figure 3. The redistribution could be evaluated and adjusted by many experts in order to achieve a situation in which the redistribution is accepted by all experts [19]. We think the consistency method is a preferred choice for the construction of consistent software certification strategy. The statistical evidence of the accuracy improvement with pairwise comparisons from approximately 15% to 5% for the one dimensional case (randomly generated bars) in [20], and from approximately 25% to 15% for randomly generated 2D shapes [21] support our expectations of improvement.

VII. Conclusions

This study demonstrates how the use of pairwise comparisons to relate such intangible software attributes as resilience

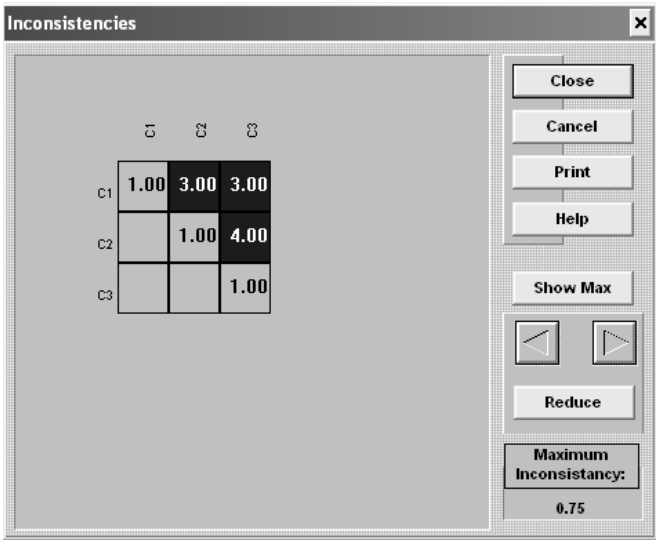

Fig. 4: Inconsistency analysis for a group with three attributes. The inconsistency is 0.75.

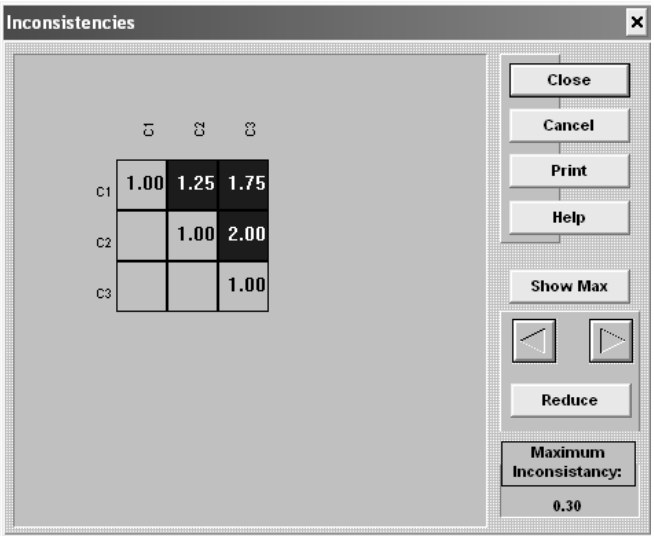

Fig. 5: Inconsistency analysis for a group with three attributes. The inconsistency is 0.3.

or reusability can result in computing weights that may be useful in establishing a software certification strategy.

Software systems are developed for different purposes. Properties such as safety, reliability and modularity could have different priorities for different projects. Software providers are obliged to provide a guarantee that their software will operate reliably, but a certification process can not imply nor guaranty that the software will not fail in all unexpected situations [22].

The desire for companies to certify their software may be driven by their ability to increase sales and to maintain a competitive advantage in the industry. This comparative advantage could be achieved if companies would develop their products while conforming to the product's requirements and industry regulations [23], [24]. A more effective software certification strategy, in terms of better resource allocation, contributes to software development cost savings. We expect that certification methods which were utilized for past projects would fit, with some minor modifications, into our software certification strategy where we use the pairwise comparisons (PC) method. This may allow for a more accurate and consistent software certification process.

ACKNOWLEDGMENTS

The authors would like to thank the Natural Science and Engineering Council of Canada (NSERC) for its partial support of this work. V. Babiy, R. Janicki and A. Wassyng also acknowledge the partial support of the Ontario Research Fund - Research Excellence.

REFERENCES

[1] J. Oh, D. Park, B. Lee, J. Lee, E. Hong, and C. Wu, *Certification of Software Packages Using Hierarchical Classification*, 2004, vol. 3026.
[2] R. Moraes, J. Dures, E. Martins, and H. Madeira, *Component-Based Software Certification Based on Experimental Risk Assessment*. Springer Berlin / Heidelberg, 2007.
[3] J. Souter, "Process certification and product testing coming together," in *Software Quality Improvement Through Process Assessment, IEE Colloquium*, Mar 1992, pp. 5/1–5/6.
[4] N. E. Fenton and S. L. Pfleeger, *Software Metrics*, 2nd ed. 20 Park Plaza, Boston, MA: PWS Publishing Company, 1997.
[5] J. K. Blundell, M. L. Hines, and J. Stach, "The measurement of software design quality," *Annals of Software Engineering*, vol. 4, no. 1, pp. 235–255, 1997.
[6] K. Lee and S. J. Lee, "A quantitative evaluation model using the iso/iec 9126 quality model in the component based development process," *Computational Science and Its Applications*, pp. 917–926, 2006.
[7] T. P. Kelly, *Improvements in System Safety*. Springer London.
[8] W. W. Koczkodaj, "A new definition of consistency of pairwise comparisons," *Mathematical and computer modelling*, vol. 18, no. 7, pp. 79–84, 1993.
[9] W. W. Koczkodaj and W. O. Mackakey, "Mineral-positional assessment by consistency-driven pairwise comparisons," *Explor. Mining Geol.*, vol. 6, no. 1, pp. 23–33, 1997.
[10] M. J. A. N. Caritat, *Marquis de Condorcet: Essai sur l'Application de L'Analyse à la Probabilité des Décisions Rendues à la Pluraliste des Voix.* Imprimerie Royale., 1972.
[11] G. T. Fechner, *Elements of Psychophysics*. Rinehart and Winston, New York, 1965.
[12] L. L. Thurstone, "Law of comparative judgements," *Psychological Review*, vol. 34, pp. 273–286.
[13] W. W. Koczkodaj, M. Orlowski, L. Wallenius, and R. M. Wilson, "A note on using a consistency-driven approach to cd-rom selection," *Library software review*, vol. 16, no. 1, pp. 4–11, 1997.
[14] R. Janicki and W. W. Koczkodaj, "A weak order solution to a group ranking and consistency-driven pairwise comparisons," *Applied Mathematics and Computation*, vol. 94, no. 2-3, pp. 227–241, 1998.
[15] A. Rae, P. Robert, and H. Hans-Ludwig, *Software Evaluation for Certification*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1994.
[16] S. Bozóki and T. Rapcsák, "On Saaty's and Koczkodaj's inconsistencies of pairwise comparison matrices," *Journal of Global Optimization*, vol. 42, no. 2, pp. 157–175, 2007.
[17] R. Janicki, "Ranking with partial orders and pairwise comparisons," *Lecture Notes in Computer Science*, vol. 5009, pp. 442–451, 2008.
[18] I. Sommerville, *Software Engineering*, 8th ed. Edinburgh Gate, Harlow Essex, CM20 2JE, England: Pearson Education Limited, 2007.
[19] W. Hasselbring and R. Reussner, "Toward trustworthy software systems," *Computer*, vol. 39, no. 4, pp. 91–92, 2006.
[20] W. W. Koczkodaj, "Statistically accurate evidence of improved error rate by pairwise comparisons," *Percept Mot Skills*, vol. 82, pp. 43–48, 1996.

[21] P. Adamic, V. Babiy, R. Janicki, T. Kakiashvili, W. W. Koczkodaj, and R. Tadeusiewicz, "Pairwise comparisons and visual perceptions of equal area polygons," *Perceptual and Motor Skills*, vol. 108, no. 1, pp. 37–42, 2009.

[22] J. Voas and K. Miller, "Software certification services: Encouraging trust and reasonable expectations," *IT Professional*, vol. 8, no. 5, pp. 39–44, 2006.

[23] F. G. Keith and I. Vertinsky, "Antecedents to certification of software development processes," in *Standardization and Innovation in Information Technology, 2007. SIIT 2007. 5th International Conference*, Oct. 2007, pp. 81–90.

[24] P. Caliman, "Software product quality evaluation and certification: the qseal consortium methodology," *http://www.cse.dcu.ie/ essiscope/sm4/qseal.doc.*, Aug. 2009.