Scalability Improvement in Software Evaluation Methodologies

Hamdy Ibrahim
Department of Electrical and Computer Engineering
University of Calgary
Canada
hibrahi@ucalgary.ca

Behrouz H. Far
Department of Electrical and Computer Engineering
University of Calgary
Canada
far@ucalgary.ca

Armin Eberlein
Department of Computer Science & Engineering,
American University of Sharjah
UAE
eberlein@ucalgary.ca

Abstract

Evaluation of software is critical in a world that increasingly relies on software. Several software evaluation methodologies have been developed, but as software solutions increases in number and size, many of them do not scale. Improving scalability of software evaluation methodologies is a challenge and failing to reach a reasonable scalability level likely constrains the adoption of an evaluation methodology. In this paper, a framework for improving scalability of software evaluation methodologies is proposed. The proposed framework relies on three keystones: categorization of evaluation criteria, dependency among criteria, and methodology adaptation. A case study is conducted to demonstrate how the proposed framework is used to improve the scalability of a hybrid evaluation model which is used to rank commercial off-the-shelf (COTS) products for a library system and select the best candidate. The case study is also used to determine which keystones are effective in improving scalability.

Keywords: Software evaluation, Scalability, COTS evaluation and selection, Agent-oriented software development.

1. Introduction

Software evaluation methodologies are invaluable when assessing several competing software products to select the best one that meets stakeholders and business needs and justifying enterprise software purchases [1]. However, the use of evaluation methodologies is usually associated with several challenges, one of which is scalability. Scalability in this context related to two aspects. The first one is related to software products and defined as a desirable property describing the ability of a software product to accommodate growth in terms of problem size and complexity. The second is related to software development/evaluation methodologies and defined as a measure of the effectiveness of a methodology when it is used in a context that is larger in scope and complexity than the context for which it was originally designed [2, 3]. Therefore, in the context of software evaluation methodologies scalability can be used as a non-functional requirement of the developed software system or as a property of software evaluation methodologies. The scalability challenge is likely to limit the adoption of software evaluation methodologies in large and complex problems (e.g., large number of criteria and candidates).

In this paper, we propose a framework for improving the scalability of software evaluation methodologies. The proposed framework uses the dependencies between evaluation criteria, criteria categorization, and methodology adaptation to improve the scalability of software evaluation methodologies. A case study is conducted to demonstrate how the proposed framework is used to improve the scalability of an evaluation methodology [4-6] used to evaluate commercial off-the-shelf (COTS) products.

The rest of the paper is organized as follows. Section 2 discusses related work. In section 3, the proposed framework that helps in improving the scalability of software evaluation methodologies is presented. Section 4 discusses how the proposed framework is used in a case study while section 5 concludes the paper and discusses possible future work.

2. Related Work

Software evaluation methodologies can be divided into two categories. The first category contains evaluation methodologies that are used to evaluate software development methodologies or processes such as those used to evaluate various agent-based, object-oriented, and agile development methodologies. The second category contains approaches used to evaluate software products (e.g., COTS evaluation and selection methodologies).

For instance, in the domain of evaluating agent-oriented development methodologies, Dam et al [7] proposed an attribute-based framework for comparing and evaluating three prominent agent-oriented methodologies: MaSE [8], Prometheus [9], and Tropos [10]. This framework focuses on four major aspects: concepts, modeling language, process, and pragmatics during the evaluation. In [11], an evaluation process is proposed to assess agent-oriented software engineering methodologies. In this evaluation process, scalability is considered as one of pragmatic criteria to represent the ability of a methodology to handle a large number of agents in an application. Further, Sabas et al [12] suggest a multi-dimensional framework containing criteria for the comparison of multiagent system methodologies. The criteria used within this framework cover the following aspects: methodology, representation, organization, cooperation, and technology.

Regarding evaluation methodologies for software products, there are numerous methods such as those used to evaluate and select commercial off-the-shelf (COTS) products [13-19]. Further, we have proposed a general purpose software evaluation (GPSE) system to help evaluators to systematically evaluate a set of software products alternatives. The GPSE system uses sound statistical methods along with the multidimensional weighted attribute framework. [20]

However, most of these evaluation methodologies consider scalability as one possible criterion against which software products are assessed. Few research efforts assess the scalability of either software development methodologies or software evaluation methodologies. So far there has been very limited effort to improve the scalability of software evaluation methodologies.

3. Scalability Improvement Framework

The proposed framework in this research relies on three keystones, shown in Figure 1, to enhance the scalability of a software evaluation methodology. The following sections discuss how each keystone contributes to the scalability improvement.

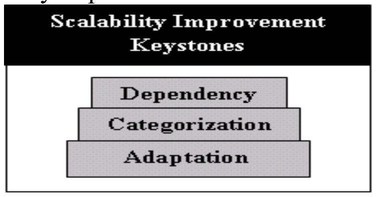

Figure 1: Framework Keystones

3.1. Dependency

We define dependency as a relationship between two or more evaluation criteria by which the change in the satisfaction of one of these criteria will result in a change in the satisfaction of other criteria that belongs to the same dependency set i.e., a set of criteria between which dependencies exist. The dependency can be used to provide information about the satisfaction level of one or more criteria using the information about the satisfaction level of other criteria belonging to the same dependency set. Therefore, dependency is used to decrease the number of evaluation criteria against which alternatives are assessed. This will reduce time, effort, and cost of conducting the evaluation process. For example, let's assume having 5 criteria that depend on each other. If an increase in the satisfaction of one criterion will increase the satisfaction of the other four criteria, then it might be sufficient to only consider this criterion during the evaluation process and its satisfaction can be used to assess the satisfaction of the other four criteria. The dependency is mathematically represented as follows:

$$C = \{c_1, c_2, ..., c_j \mid where \ : c_j = criterion \ "j"\}$$

$$D = \{R_1, R_2, ... R_i\}$$

$$R_i = \{(c_a, c_b), relation\_desc\}$$

Where:

C: a set of evaluation criteria.

D: a dependency set that consists of a number of relations sets.

R_i: a subset of CXC. It represents any binary directed relationship between two evaluation criteria. For example, the pair (c_a,c_b) means a relationship existing between c_a and c_b and a directed link from c_a to c_b.

The dependency has the following forms:

1. The existence of one criterion is completely based on the existence of at least one other criterion. This means the satisfaction of this criterion by default requires the satisfaction of other criteria on which it is based.
2. Direct versus indirect dependency. Direct dependency means that the criteria are directly related to each other. However indirect dependency is a relation between criteria through other criteria. For example, if there is a direct relation between (c1, c2) and between (c2, c5), then there is an indirect relation between (c1, c5).
3. Unidirectional versus bi-directional dependency. Unidirectional refers to a relation where criterion A depends upon criterion B. Therefore, a change in B implies a potential or possible change in A. However a bi-directional dependency refers to a relation where the two criteria depend upon each other. Therefore, a change in any criterion implies a change in the other.
4. Linear versus nonlinear dependency. Linear is the simplest form of relation and it means that the satisfaction of a criterion changes linearly with the change in the satisfaction of another criterion.

Dependency analysis aims at identifying relationships between evaluation criteria as well as their types. Therefore, for each criterion, we need to determine if it has any relationships other criteria, what the relationship types are, and how the satisfaction of the criterion is determined using the relationships and information about the satisfaction of criteria involved in the relationships. Figure 2 shows activities performed during dependency analysis.

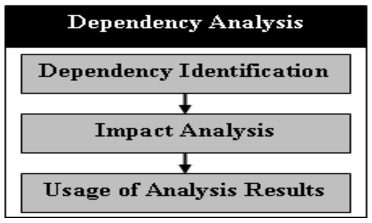

Figure 2: Dependency Analysis Activity

Dependency analysis has three phases. The first phase is dependency identification during which a list of relationships that may exist between criteria as well as the criteria involved in these relations are identified. Dependency graphs along with dependency tables are used to represent and keep track of the dependency relationships between evaluation criteria. Table 1 shows an example for the dependency table and Figure 3 shows the corresponding dependency graph. As shown in Figure 3, the dependency graph consists of a set of nodes representing evaluation criteria, arcs representing the dependency between them, and label r_{ij} describing the relation. In the example given in Table 1 and Figure 3, there is a bi-directional relation between C_1 and C_2. However, a unidirectional relation exists between (C_1, C_j) and (C_j, C_2).

Table 1: Dependency Table

Criterion	C_1	C_2	...	C_j
C_1		r_{12}		r_{1j}
C_2	r_{21}			
...				
C_j		r_{j2}		

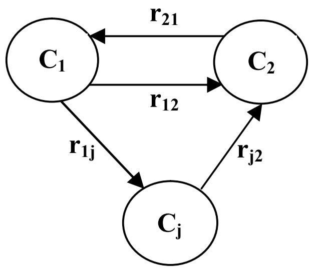

Figure 3: Dependency Graph

In the second phase, called impact analysis, the identified relations are analyzed to determine their types and their impact on the involved criteria. In the third phase, the results of the impact analysis phase are used to determine how the relation and information about the satisfaction of some criteria is used to assess the satisfaction of other criteria in the same dependency set.

3.2. Categorization

Categorization or classification is the process of dividing or distributing the evaluation criteria into classes or categories based on a shared attribute(s). In categorization, the evaluation criteria are also assigned to pre-defined categories. It offers the following benefits:

1. An easier and better way to understand the criteria and how they are related to each other.
2. Categorization assists evaluators in handling the criteria easily.
3. Classification facilitates the inference about the satisfaction of one criterion using the information about the satisfaction of another criterion and the relation between them.
4. Evaluators are assigned to categories that match their experiences.
5. Evaluators focus on a small number of evaluation criteria. This increases their productivity and speeding up the overall evaluation process.

There are several methods for categorizing the criteria. The first method categorizes the evaluation criteria based on the type of criteria. For example, the technical criteria representing functional and non-functional requirements are considered as a category. However, the non-technical criteria such as license, contract, and vendor related issues can be considered as another category. Furthermore, the technical criteria category can be divided into two sub-categories: functional criteria and non-functional criteria. Another way is to set up a category for criteria related to each other or criteria with shared attributes such as the ones related to vendors can be in the same class.

3.3. Adaptation

Adaptation of an evaluation methodology refers to the process of configuring, customizing, or tailoring the methodology to fit better with the specific project characteristics and needs. It primarily creates a methodology instance adapted to the characteristics and constraints of the project undertaken in a particular organization. The motivation behind the use of methodology adaptation is that it is rare to find a methodology that is applicable and suitable to every project. This is because each project has its own unique characteristics and constraints (e.g., business process, number of evaluation criteria, available resources in terms of budget, time, and workforce allocated to the project). Moreover, the adaptation is a recommended strategy for improving the scalability and applicability of software evaluation methodologies [21]. Figure 4 shows how the evaluation methodology is adapted. Briefly, there is a list of optional activities/tasks and roles from which a list of recommended activities/tasks and roles that fit with the project features and constraints are selected. Examples for the options, determined during the methodology adaptation, are as follows:

- The activities/tasks that are performed.
- The roles that are involved in the evaluation process.
- The execution depth of each task.
- The number of iterations that will be executed.
- The techniques that are applied in each step during the evaluation process.
- The evaluation strategy that is used.
- The documentation method that is used.

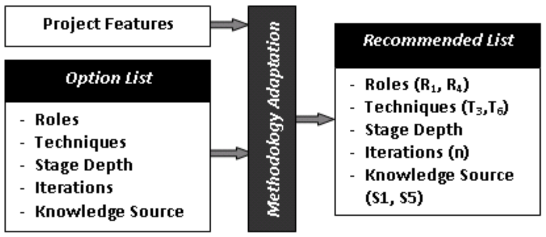

Figure 4: Methodology Adaptation Keystone

4. Case Study: COTS Evaluation and Selection

COTS-based development (hereafter CBD) is defined as the development of new software systems using COTS products [22]. CBD has received much attention because it offers potential benefits like reduction of development time and effort, improvement of the quality of the final software product, and offering functionality to stakeholders which might not have been requested initially but which is still beneficial [23]. CBD includes activities such as search and evaluation criteria definition, COTS evaluation and selection, COTS adaptation and integration. COTS evaluation and selection is considered as an essential task in CBD. [24]. Moreover, we will apply the proposed scalability improvement framework on a hybrid evaluation model proposed in [4-6] to check the impact of the keystones on the scalability of the evaluation approach. The hybrid evaluation model evaluates and ranks COTS candidates and helps decision makers to select the best candidate.

4.1. A Hybrid Evaluation Model for Ranking COTS Products

We have proposed a hybrid model, composed of Bayesian Belief Network (BBN) and Analytic Hierarchy Process (AHP), to rank various COTS candidates and select the best candidate meeting stakeholder needs while explicitly considering uncertainty in terms of incomplete, inaccurate, and inconsistent information about COTS candidates. The BBN model primarily addresses the problem of having incomplete and inaccurate information about COTS candidates. As shown in Figure 5, AHP does not belong to either COTS evaluation or uncertainty management because AHP is used during the evaluation process to rank various candidates as well as to address inconsistency possibly existing in COTS candidates' information during the uncertainty management. This is why AHP is kept outside the COTS evaluation and uncertainty management blocks. The model has two outcomes; the first, called the satisfaction value, is created by AHP and represents how much each candidate satisfies the evaluation criteria and the second, called the confidence level, is created by the BBN model and represents the probability that the candidate's satisfaction is high. The confidence level represents our confidence regarding the satisfaction level of various candidates. The overall model is a step towards conducting the COTS evaluation and selection process explicitly considering uncertainty. The process of using the model is shown in Figure 5 and more details about the model and its usage can be found in [4-6]. The hybrid evaluation model was used to evaluate and rank various COTS candidates used to automate a Library system while explicitly representing uncertainty.

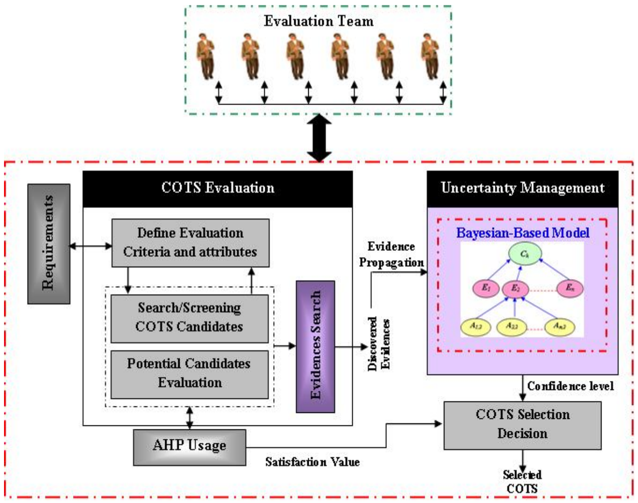

Figure 5: The usage process of the hybrid evaluation model

4.2. Case Study: Results and Discussions

Three people (i.e., two have experience in library science and automation and the third has experience in software development) participated in the case study. We consider four cases during the use of the proposed framework to improve the scalability of the hybrid evaluation model. Further, we examine the effectiveness of the dependency and categorization keystones and their impact on the overall time of the evaluation process. The adaptation keystone is not considered since the number of evaluation criteria and number of COTS candidates are relatively small (i.e., 21 criteria and 4 candidates) and based on the experience of the participants, there will be no big difference whether the proposed framework is adapted or not. The four cases considered during the case study are as follows:

1. ***Base case***

In this case, the hybrid evaluation model is used without considering the dependency between criteria and without categorizing the criteria.

2. ***Dependency case***

In this case, the hybrid evaluation model is used considering the dependency between criteria. In this case, the overall time includes the time consumed to perform dependency analysis and the evaluation.

3. ***Categorization case***

In this case, the hybrid evaluation model is used after dividing the criteria into three categories. The three categories are the technical criteria related to the functionality of the library system (there are 10 criteria in this category), the technical criteria related to non-functional requirements as performance, security, and maintainability (there are 5 criteria in this category), and the non-technical criteria related to license, contract, and vendor capability and reputation issues (there are 6 criteria in this category). Each evaluator is assigned to a category that matches his experience. In this case, the overall time includes the time consumed to perform categorization and the evaluation.

4. ***Dependency and categorization case***

Finally, the hybrid evaluation model is used considering both dependency and categorization. The criteria are categorized first then the dependencies between criteria existing in the same category (i.e., it is usually called internal dependency) are analyzed and their impact are identified.

Figure 6 shows the change in the time required to conduct the evaluation process as the number of criteria changes in cases 1, i.e., the base case, and 2, i.e., the dependency case. Obviously, increasing the number of criteria in both cases increases the overall time of the evaluation. However, the overall time in the dependency case is less than the overall time in the base case. Therefore, considering the dependency between the criteria decreases the overall time required to conduct the evaluation process, i.e., improving the scalability of the evaluation model. Further, the graph is not linear because the time required to evaluate the candidates against each criterion is different than the time required to evaluate the candidates against other criteria.

Figure 7 shows a comparison between the four cases with respect to the overall time of the evaluation process. Table 2 shows the reduction percentage in the overall time of the evaluation in the based case when dependency, categorization, or dependency and categorization are considered. Table 2 shows that considering the dependency between the criteria, categorizing the criteria, or both of them reduces the overall time 24%, 31%, or 41% respectively. This means that the use of the dependency and categorization keystones is more effective than using only one of them.

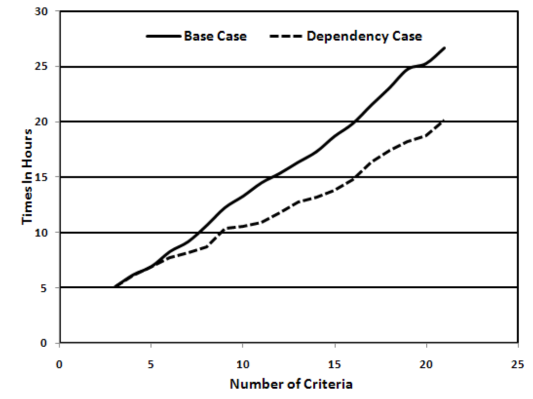

Figure 6: Change in the Overall Time in Cases 1 and 2

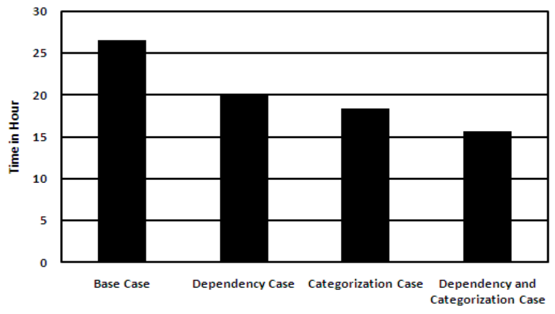

Figure 7: The Overall Time in the Four Cases

Table 2: Reduction Percentage in Overall Time

Case	1	2	3	4
Overall Time in Hours	**26.6**	**20.2**	**18.4**	**15.6**
Reduction %		**24%**	**31%**	**41%**

5. Conclusion and Future Work

In this paper we have proposed a framework to improve the scalability of software evaluation methodologies. The proposed framework is used to improve the scalability of a hybrid evaluation model used to evaluate and rank commercial off-the-shelf (COTS) products and to select the best candidate for automating a library system. Based on the results obtained, we can conclude that:

1. Improving the scalability of software evaluation methodologies enables the use of these methodologies in a context that is larger in scope or complexity than that for which it was designed because the scalability improvement saves resources in terms of time required to conduct the evaluation process.
2. Each keystone (e.g., dependency and categorization) contributes to the scalability improvement. However,

one keystone might be more effective than other keystones (e.g., categorization is more effective than dependency and the use of both keystones is more effective than the use of only one).
3. The case study shows that the use of dependency and categorization keystones reduces the overall evaluation time 41%. However, it is necessary to apply the proposed framework in several case studies to generalize these results.

Further research is needed in the following areas:

1. Checking the effectiveness of the proposed framework when used to improve the scalability of software evaluation methodologies in case of large problems with a large number of criteria and candidates as well as a large number of evaluators.
2. Effectiveness of evaluation methodology adaptation needs further investigation because the methodology has not been adapted in the presented case study.
3. Our focus was on internal dependencies. However, the dependencies between criteria that exist in different categories needs to be considered. In that case, the impact of communication between evaluators to exchange information about the criteria will be considered (i.e., there is a communication overhead).

References

[1] G. Gediga and K. Hamborg, "Evaluation of Software Systems", *Encyclopedia of Computer Science and Technology*, Vol. 45, 2001, pp. 166-192.

[2] E.J. Weyuker and A. Avritzer, "A Metric to Predict Software Scalability", *Proc. of the 8th Int. Symposium on Software Metrics*, IEEE Computer Society Washington, DC, USA, 2008, pp. 152-158.

[3] L. Duboc, D.S. Rosenblum, and T. Wicks, "A framework for modelling and analysis of software systems scalability", *Proc. of the 28th int. conf. on Software engineering,* ACM New York, NY, USA, 2006, pp. 949-952.

[4] H. Ibrahim, B.H. Far, A. Eberlein, "Towards Uncertainty Management in COTS-Based Development", *Proc. of the 9th IASTED Int. Conf. on Software Engineering and Applications SEA 2008*, Orlando, Florida, USA, 2008, pp. 133–140.

[5] H. Ibrahim, B.H. Far, A. Eberlein, "Weighted criteria hybrid model for ranking commercial off-the-shelf products", *Proc. of the IASTED Int. Conf. on Software Engineering SE 2009*, Innsbruck, Austria, 2009, pp.49-56.

[6] H. Ibrahim, B.H. Far, A. Eberlein, "Tradeoff and Sensitivity Analysis of a Hybrid model for Ranking Commercial Off-The-Shelf Products", *Proc. Of the 16th Annual IEEE Int. Conf. and Workshop on the Engineering of Computer Based Systems (ECBS)*, San Francisco, CA, USA, 2009.

[7] K. Dam, and M. Winikoff, "Comparing Agent-Oriented Methodologies", *Proc. of the 5th Int. Bi-Conf. Workshop on Agent-Oriented Information Systems (AOIS'03)*, Melbourne, Australia, 2003, pp. 78-93.

[8] Scott A. DeLoach, "Analysis and Design using MaSE and agentTool", *Proc. of the 12th Midwest Artificial Intelligence and Cognitive Science Conf. (MAICS 2001),* Miami University, Oxford, Ohio, March 31 - April 1, 2001.

[9] L. Padgham and M. Winikoff, "Prometheus: A Methodology for Developing Intelligent Agents", *Proc. of the First Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 2002)*, 2002.

[10] F. Giunchiglia, J. Mylopoulos, and A. Perini, "The tropos software development methodology: processes, models and diagrams", *Proc. of the first int. joint conf. on Autonomous agents and multiagent systems: part 1*. Bologna, Italy, 2002.

[11] C. Lin, K. M. Kavi, F. T. Sheldon, K. M. Daley and R. K. Abercrombie, "A Methodology to Evaluate Agent Oriented Software Engineering Techniques", *Proc. of the 40th Hawaii Int. Conf. on System Sciences, IEEE Computer Society*, Washington, DC, USA , 2007, pp. 60-69.

[12] A. Sabas, M. Badri, and S. Delisle, "A Multidimentional Framework for the Evaluation of Multiagent System Methodologies", *Proc. Of the 6th World MultiConf on Systemics, Cybernetics and Informatics (SCI-2002)*, 2002, pp. 211-216.

[13] J. Kontio, "A Case Study in Applying a Systematic Method for COTS Selection", *Proc. of ICSE-18*, 1996, pp. 201–209.

[14] B. P. Cavanaugh and S. M. Polen, "Add Decision Analysis to Your COTS Selection Process", *The Journal of Defense Software Engineering*, April 2002.

[15] C. Alves and J. Castro, "CRE: A Systematic Method for COTS Selection", *Proc. Of XV Brazilian Symposium on Software Engineering*, Rio de Janeiro, Brazil, 2001.

[16] C. Ncube and N.A.M. Maiden, "Guiding Parallel Requirements Acquisition and COTS software", *Proc. of IEEE Int. Symposium on Requirements Engineering*, 7-11 June 1999, pp. 133 – 140.

[17] D. Kunda, "A social-technical approach to selecting software supporting COTS-Based Systems", *PhD Thesis*, Department of Computer Science, University of York, Oct. 2001.

[18] X. Burgues, C. Estay, X. Franch, J. A. Pastor, and C. Quer, "Combined Selection of COTS Components", *Proc. Of The First Int. Conf. on COTS-Based Software Systems (ICCBSS'02)*, Orlando, Florida, 2002, pp. 54-64.

[19] L. Chung, K. Cooper, and S. Courtney, "COTS-Aware Requirements Engineering: The CARE Process", *Proc. of the 2nd Int. Workshop on Requirements Engineering for COTS Components*, Kyoto, Japan, September 7, 2004, available at: http://www.lsi.upc.edu/events/recots/home.html

[20] B.H. Far, V. Mudigonda, and H. Elamy, "A General Purpose Software Evaluation System", *Proc. Of IEEE Int. Conf. on Information Reuse and Integration (IRI)*, 2008, pp. 290-295.

[21] R.L. Glass, "Matching methodology to problem domain", *Communications of the ACM*, Vol. 47, Issue 5, 2004, pp. 19-21.

[22] X. Franch and M. Torchiano, "Towards a Reference Framework for COTS-based development: a proposal," *Proc. of the second int. workshop on models and processes for the evaluation of off-the-shelf components, Int. Conf. on Software Engineering*,2005, pp. 1-4.

[23] S.B. Sassi, L.L. Jilani, and H.H.B. Ghezala, "Towards a COTS-Based Development Environment", *Proc. Of the 5^{th} Int. Conf. on Commercial Off-The-Shelf (COTS)-Based Software Systems (ICCBSS 2006)*, pp. 167-176.

[24] P. Oberndorf, "Facilitating Component-Based Software Engineering: COTS and Open Systems," *Proc. of the 5th Int. Symposium on Assessment of Software Tools – "SAST'97"*, pp.143-148.