A Framework for Quality Evaluation of Tools in Software Product Lines

Muhammad Garba
Department of Computer Science
Kebbi State University of Science and Technology, Aliero, Nigeria
garbamga@gmail.com

Muhammad Sirajo Aliyu
Department of Computer Science
College of Science and Technology,
Hassan Usman Katsina Polytechnic
Katsina State, Nigeria,
muhammadsirajo@yahoo.com

Abstract -- *Software quality has become a key aspect of good software engineering practice. The term quality is a complex concept. Since it means different things to different individuals, it is highly context-dependent. Just as there is no single mobile phone to satisfy everyone's needs, likewise there is no universal definition of quality. Accordingly, there can be no one, simple measure of software quality acceptable to everyone. This paper presents a benchmark for evaluating quality attributes important for practical use of software product line (SPL) tools. The benchmark focused on measuring the four quality attributes: Usability, Performance, Scalability, and Integration. The results are to assist practitioners and researchers alike by providing a standard and empirical approach to evaluating product line tools in the future. It also identifies and recommends areas that need attention in future tools design in this kind of modelling.*

Keywords-Benchmark, Software Quality, Evaluation, Feature Modelling, Software Product Line Variability Models

I. Introduction

The Software Product Line Engineering (SPLE) technique provides a systematic way to reuse software assets. These assets are the software artefacts or resources associated with your products. The artefacts include, but are not limited to requirements analysis, design specifications, software implementation, configuration, test plans, test cases, etc. The assets are then engineered to be shared across the entire product line, i.e., to be used in multiple products. Therefore, SPLE is a technique that optimizes the reuse of existing software assets by creating multiple applications that share many features, while still exhibiting certain differences [1, 2].

The main purpose of software reuse is to improve software quality and productivity. Software reuse is of interest because people want to build systems that are bigger and more complex, more reliable, less expensive and that are delivered on time. They have found traditional software engineering methods inadequate, and feel that software reuse can provide a better way of doing software engineering [3]. However, Well-designed metrics with documented objectives can help organizations obtain the information it needs to continue to improve its software product, processes, and customer services. Therefore, future research is need to extend and improve the methodology to extend metrics that have been validated on one project, using our criteria, valid measures of quality on software product line projects.

This paper is a summary of an extended paper which presents a benchmark for evaluation of quality attributes important for practical use in software product lines. The purpose is to provide the researchers and the practitioners with a better insight into the validation activity, improving the software process towards the goal of the having a management process. The study identified and selected 10 product line variability management tools based on their availability and support for feature models, to be evaluated using the benchmark in order to identify whether and to what extent these tools provide support for the identified quality attributes. The quality attributes studied in the evaluation are: usability, performance, scalability and integration, which are seen as being important for the practical use of these tools.

The remaining paper is organised as follows: In section 2, the research methodology used is discussed. Section 3 provides the detailed description of the Benchmark. Finally, Section 4 presents the related works before section 5 rounds off the paper with the conclusion.

II. Related works

Many works have been reported by various authors within the SPL community in order to analyse, compare, or evaluate some of the existing variability management methods, tools, and techniques. However, to the best of our knowledge, no one has specifically evaluated these quality characteristics important for practical use of tools that support variability in SPLs.

For example, in [15], a quality evaluation of nine feature modelling tools was conducted with the specific focus on quality criteria of usability, safety, and functional usability features. The main aim of the investigation was how to improve the quality in feature modelling tools, in general. Study [16] evaluated four product line tools against certain criteria defined based on three perspectives; 1) criteria relating to product line engineering (2) criteria relating to tools capabilities and (3) criteria concerning project management. This is to determine their ability to satisfy industry expectations. In study [17], eight tools and techniques for variability modelling in software product line (SPL) or business process management (BPM) were evaluated based on various formalisms used in specifying software process variability. The study analysed the tools in order to investigate their suitability for modelling variability in the software process. However, in order to assist engineers in selection of a suitable tool that best fits their needs, the authors in [18] conducted an exploratory study that compares and analyses two feature modelling tools, based on data collected from 56 participants who experimentally used the tools. The study focused on evaluating the four common functionalities provided by feature modelling tools. These are: feature model editor, automated analysis of feature model, product configuration and tool notation.

III. Methodology

In order to carry out this study, we applied a research methodology that combined both the features of qualitative and quantitative research methodologies. In the first step, a benchmark was developed, to be used consistently as a guideline in the evaluation process. As a crucial stage in the benchmarking design, we explored product line industries in order to know precisely what matters for the practitioners. We, therefore, used the outcome of an interview-based survey that involved a number of software product line practitioners, in which they were asked to list five quality attributes they deemed important for practical use of SPLs Variability Management (VM) tools. The identified quality attributes (usability, scalability, performance, and integration) were then used as key criteria to assess (i.e., how well the tools addressed them) the capability of SPLs-VM tools in the evaluation phase. Details of these quality attributes are given in section 4.1.

In the second step, the study focused on measuring the identified quality attributes, so as to ascertain their meanings and position. Hence, a further exploration into a number of internationally recognised standards and some respected reference models were carried out; these included ISO/IEC 9126 [4, 5] (International Standard for Evaluation of Software Quality) and IEEE Standard 610.12 (IEEE Standard Glossary of Software Engineering Terminology). Among the other is the well-known Software Quality Metrics book [6], as well as An Effort-Based Framework for Evaluating Software Usability [7].

Having completed the survey and investigations on the identified quality attributes, in the third step, the results of a study has been used, this study reported on a survey in which 37 existing product line-variability management tools were identified and analyzed using a systematic literature review [8], from which 8 tools were selected (cf. extended paper), based on their availability and support for the graphical notations. However, 2 more publicly available tools were added using a separate search, making a total of 10 tools used in the evaluation process. The details of the identified tools and the criteria used when selecting a tool are given in the extended paper of this work.

Finally, in the fourth step, an experimental evaluation was conducted, using 4 sample case studies (cf. extended paper) of different sizes, and this was achieved by steadily applying the benchmark. The purpose was to assess how well the identified tools addressed the four quality attributes. This was followed by an opinion-based evaluation method that uses a questionnaire to obtain more insight into the user's opinion of their experience using the system. This was to know the extent to which the system is attractive.

IV. Benchmark

This section presents the four quality attributes measured, sub-characteristics of each quality attribute and their detailed definitions. The section also gives in detail, how the measurement was carried out.

A. Quality Attributes

The four quality attributes this study measured are: usability, scalability, performance, and integration. As stated in section 2, these attributes were gathered from a study that used an interview based survey involving a number of software product line practitioners, in which they were asked to list five most important quality attributes for practical use of SPL tools. Figure 1. depicts the four quality attributes with their sub-characteristics.

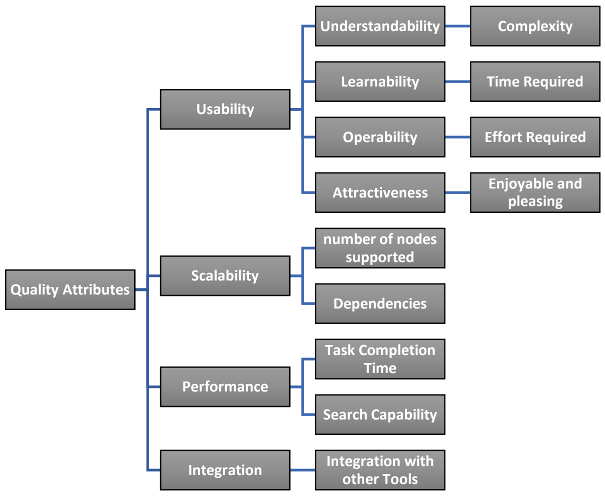

Fig. 1. The quality attributes used

B Usability Measure:

In order to determine and understand the main aspects that influence usability, this study based the measurement on the ISO 9126 [4, 5] on software quality and measurement, which defined usability as 'the capability of the software to be understood, learned, used and liked by the user, when used under specified conditions'. The standard identifies four to five key components of usability of a software product. Below are the detailed breakdown and the definitions of these sub-quality characteristics of usability:

C Understandability

Can the software be understood easily? That is, the ability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and given the conditions of use. Understandability helps determine how easily the user can comprehend and use the software. We based the measurement of Understandability on study [9] where an ordinal scale was used as our measurement scale type (see Table 1) to measure the complexity of using the software. The ordinal scale provides a list of ordered alternatives from which respondents can select an option.

TABLE 1 ORDINAL SCALE TYPE

Value	Meaning
1	Trivial: commonly encountered (no exceptional effort needed)
2	Simple: Easy to manage and uncomplicated
3	Moderate: Being within average limit
4	Complex: Not easy to manage of being intricate
5	Incomprehensible: Impossible to manage of being not clear

D Learnability

Can the software be learnt easily? That is, the ability of the software product to enable the user to learn its application. Learnability is measured as the time that is required to fulfil a specified task. The specified task for this study is the need to add, delete, and edit a feature. This is in addition to the modelling of its dependency.

Learnability = Total Time required to **Add**, **Delete** or **Edit** a feature + **Dependency** Management

E Operability

Can the software be operated with minimal effort? That is, the capacity of the software product to allow the user to operate and control it. Operability was measured based on the efforts needed to accomplish the specified tasks (in this case) of adding, deleting, and editing a feature, together with the modelling dependency. Consequently, this effort equals the number of mouse clicks or screen touch (mc/st) + number of keyboard hits (kh). This measurement method is based on [7].

Operability = **Efforts needed** to **Add**, **Delete** or **Edit** a feature + **Dependency** Management

Efforts = Number of mouse click or equivalent + Number of Keyboard strikes

F Attractiveness

Is the interface of the software engaging? That is, the capability of the software product to be liked by the user. To measure attractiveness, this study based on [9] where a 5-point Likert scale is used to rank the software attractiveness, given a user a statement with which the user agrees or disagrees. The statement used for this study is: The software is attractive (i.e. Enjoyable and pleasing).

1- Strongly Agree 2- Agree 3- Neither agree nor disagree 4- Disagree 5- Strongly Disagree

G Compliance

Does the software meet existing usability standards? From the above definitions, usability can be measured by the degree to which a software product can satisfy the individual aspects of the definitions, i.e. to learn, understand, operate, and be attractive, while at the same time the software is compliant with and meets the existing usability standards. This is to be achieved under specified conditions in which a user or group of users carry out certain practical tasks.

H Basics of sub-quality attributes under usability

i. Understandability: Complexity in using the software
ii. Learnability: Time required to fulfil a specified task
iii. Operability: Effort required to carry out a basic task
iv. Attractiveness: Is the software attractive to the target audience?

I Scalability Measure:

Scalability, as it has been defined by [10], is the ability of the modelling approach to continue to meet its throughput objectives despite increasing or decreasing the amount of assets and elements that make up the models. A scalable variability modelling approach is the one that is useful when applied to a product line of any size (i.e. it should be capable of managing large or small size variability without any overhead or extra effort). Therefore, an approach will not be regarded as scalable if scaling only in one direction (i.e. downwards or upwards). However, a survey study on scalability aspects in [11] pointed out that, dependency relationships (such as variants to variants, variants to variation points or variation points to variation points) within variability models are the most discussed aspects in tackling scalability by modelling approaches. Hence, based on these studies, we used sample case studies of various sizes to serve as our basis for the experimental process of measuring scalability. These cases were then classified into three different categories, which were then used to validate the selected tools with respect to this quality aspect. The sample models are: (1) Small size, when a tool supports the development and management of 10-50 features before it starts to freeze or slow down. (2) Medium size, when the ability of variability management tool is to offer support for the development and management of 10-100 features when used, and (3) Large size, when it supports the development and management of variability models between 100-1000. At each level of testing of these various sample models, there was a practical investigation to see if the tools provide good support for dependency management and how it works. The scalability measure has been achieved experimentally, in order to gain a clear understanding of how and to what level the selected tools offer quality support for this attribute during the modelling process. Please note that it is not our purpose to measure the visualization techniques deployed by these tools, but rather focus on the number of nodes they support.

V Performance Measure:

Performance evaluation according to [12] and [13] includes externally observable system performance characteristics, such as response times and completion rates. However, IEEE standard 610.12 defined performance as the degree at which a system or a component completes designated tasks within given limits, such as speed, accuracy, or memory usage [14]. In this study, Performance is measured in relation to the scalability as the time it takes for each tool to validate the sample feature models assigned to it. That is, performance is measured as task completion time plus the search capability provided by the tools. Due to a large growth in size of the model, it becomes mandatory to investigate whether a tool can allow its user to search for a particular element of interest given several features.

Integration Measure:

The ability of a software tool to provide the means to either fully or partially integrate with other tools so that both tools can operate on the same set of data.

In this context, we will be using characteristics as follows:

Y = Yes, when a tool provides means to be fully integrated with other tools, and therefore operate on same set of data.

P = Partial, when a tool provides only half the features required for integration.

N = No, when a tool provides no means of integration.

VI Conclusion

This paper presents a description of a benchmark for evaluating quality attributes important for practical use of software product lines (SPLs). The focus was on measuring four quality attributes: Usability, Performance, Scalability, and Integration. It however, reported briefly on how empirical experimentation can be conducted if involves a number of variability management tools based on their availability and support for feature modelling. The paper also describes the importance of determining and gaining a detail understanding of where and how the quality of variability management tools could be improved during evaluation process, and this is to get a clear understanding on whether and to what extent these tools provide support for the identified quality attributes. The paper also highlighted the importance of using multiple case studies of various sizes and data elements while conducting evaluation of this kind. In addition, one of these cases should be a real live data, either acquired from industry or any related software organization. Meanwhile, the remaining case studies could be gathered from some results of a careful examination of a large body of research in the area of software product lines, from which feature models of various sizes could be formulated and used in the experimental process.

Reference

[1] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*. Massachusetts: Addison-Wesley, 2002.

[2] K. C. Kang, J. Lee, and P. Donohoe, "Feature-Oriented Product Line Engineering," *IEEE Software*, vol. 19, pp. 58-65, 2002.

[3] L. Chen and M. Ali Babar, "A systematic review of evaluation of variability management approaches in software product lines," *Information and Software Technology*, vol. 53, pp. 344–362, 2011.

[4] ISO/IEC, "ISO/IEC 9126-1: Software Engineering-Product Quality-Part 1: Quality Model," ed. Geneva Switzerland: International Standards Organization, 2001.

[5] ISO/IEC, "ISO/IEC 9126-1: Software Engineering-Product Quality, Part-2, External Metrics," ed. Geneva, Switzerland: International Organization for Standardization, 2003.

[6] N. Fenton and J. Bieman, *Software Metrics: A Rigorous and Practical Approach*, 3 ed.: CRC Press, 2014.

[7] D. Tamir, C. Mueller, and O. Komogortsev, "An Effort-Based Framework for Evaluating Software Usability Design," *ARPN Journal of Systems and Software*, vol. 3, pp. 65-77, 2013.

[8] R. Bashroush, M. Garba, R. Rabiser, I. Groher, and G. Botterweck, "CASE tool support for variability management in software product lines," *ACM Computing Survey*, vol. 50, p. 45, March 2017 2017.

[9] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, 2 ed. Boston, USA: PWS Publishing Company, 1998.

[10] K. Berg, J. Bishop, and D. Muthig, "Tracing software product line variability: from problem to solution space," in *Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, South Africa, 2005, pp. 182 - 191.

[11] L. Chen and M. A. Babar, "A Survey of Scalability Aspects of Variability Modeling Approaches," in *Workshop on Scalable Modeling Techniques for Software Product Lines at SPLC*, 2009.

[12] D. Ferrari, *Measurement and Tuning of Computer Systems*. New York: Prentice Hall, 1983.

[13] L. Kleinrock, *Queueing Systems, Volume I: Theory, and Volume 2: Computer Applications*. New York: Wiley, 1976.

[14] IEEE, "IEEE Standard Glossary of Software Engineering Terminology," in *IEEE Std 610.12-1990*, ed, 1990, pp. 1-84.

[15] M. El Dammagh and O. De Troyer, "Feature Modeling Tools: Evaluation and Lessons Learned," in *Proceedings of the 30th International Conference on Advances in Conceptual Modeling: Recent Developments and New Directions (ER'11), Variability Workshop on Software Variability Management (Variability@ER11)*, 2011, pp. 120-129.

[16] O. Djebbi, C. Salinesi, and G. Fanmuy, "Industry Survey of Product Lines Management Tools: Requirements, Qualities and Open Issues," in *Proceedings of the 15th IEEE International Requirements Engineering Conference (RE '07)*, New Delhi, India, 2007, pp. 301-306.

[17] J. Simmonds, M. C. Bastarrica, L. Silvestre, and A. Quispe, "Analyzing Methodologies and Tools for Specifying Variability in Software Processes," Universidad de Chile, Santiago, Chile. 2011.

[18] J. A. Pereira, C. Souza, E. Figueiredo, R. Abilio, G. Vale, Heitor, *et al.*, "Software Variability Management- An
Exploratory Study with Two Feature Modeling Tools," in *VII Brazilian Symposium on Software Components,
Architecture and Reuse*, 2013.