Analyzing the Importance of Learnability and Understandability Quality Attributes in Reference to SPL Feature Models

Geetika Vyas
Dept. of CS & IT
The IIS University
Jaipur, India
geetika.vyas@iisuniv.ac.in

Dr. Amita Sharma
Dept. of CS & IT
The IIS University
Jaipur, India
amita1983@iisuniv.ac.in

Astha Pareek
Dept. of CS & IT
The IIS University
Jaipur, India
astha.pareek@iisuniv.ac.in

Abstract— The modeling foundation of SPL is promoting software reuse by segregation of variant features of all the products which belong to a family. The analysis of various quality attributes is very important in reference to SPL feature models. Identifying whether a feature model is easy to use and learn will help develop a successful product line. Two important quality sub factors of usability i.e. understandability and communicativeness play a great role in development of successful product line feature model. If the understanding of any feature model is low, it will result in lesser use of that feature model. Same applies to communicativeness, the more the communicativeness of a feature model, the more the usability. In other words, the successful reuse of any feature model will depend on the degree of its understanding and communicativeness. Current analysis methods usually focus only on functional requirements of the product lines and do not focus on product quality. Whereas, non functional requirements like maintainability, dependability and usability etc are essential dimensions of variability. This paper is intended to study the role of understandability and communicativeness over feature models. It also throws light on the effect of these quality sub factors on SPL feature models and suggests ways to improve their degree.

Keywords *—quality, software product line, usability, learnability, understandability, communicativeness.*

*****

.

I. INTRODUCTION

A growing trend in software development is the requirement to develop multiple and similar software products instead of just a single individual product. There are several reasons behind this. Products being developed for the international market must be adapted for diverse cultural or legal environments, and for different languages, and so must provide appropriate user interfaces. Due to cost and time constraints it is not possible for software developers to develop a new product from scratch for each new customer, and so software reuse has to be increased. Such types of problems are typically seen in portal or embedded applications, e.g. vehicle control applications. Software Product Line Engineering (SPLE) offers a solution to these not so new, but increasingly challenging problems. The demand of improved software quality is increasing at rapid pace. Measuring quality at early phase of software development is the key to develop high-quality software product line. Software product line feature models should be so developed that they are easily understandable, testable, and modifiable. Quality attributes are the overall factors that affect run-time behavior, system design, and user experience. Some of these attributes are related to the overall system design, while others are specific to run time, design time, or user centric issues. The extent to which the software product line possesses a desired combination of quality attributes such as usability, performance, reliability, and security indicates the success of the design and the overall quality of the software product line. Out of these usability plays special role in SPL feature models.

This work focuses on the usability assessment of software product line feature models, primarily focusing on understandability and communicativeness of the same.

Section II holds introduction about software product line. Section III talks about feature oriented programming. Section IV briefs about what are features and feature models. Section V focuses on software quality attributes. Section VI analyses the available metrics and section VII concludes the whole paper.

II. SOFTWARE PRODUCT LINE ENGINEERING

Products being developed for the international market must be adapted for diverse cultural or legal environments, and for different languages, and must provide appropriate user interfaces. Due to cost and time constraints it is not feasible for software developers to develop a new product from scratch for each new customer, and so software reuse has to be increased.

Software Product Line Engineering (SPLE) offers a way out to these not so new, but increasingly challenging problems [1].

SPLE is an approach that develops and maintains families of products keeping track of their common aspects and predicted variability's at the same time. SPLE focuses on reuse and is a viable and important software development paradigm [2].

III. FEATURE ORIENTED PROGRAMMING

Feature Oriented Programming was designed keeping an eye towards SPLE. The basic idea is to decompose a software system in terms of the features it provides i.e. to modularize the software system into feature model which is a representation of product features and the dependencies of these features. It is used explicitly for systematically defining the commonality and variability [3]. This methodology is recommended because of its ability to produce numerous similar but functionally different programs from the same set of features [4]. This is achieved by simply selecting the desired features. It helps in engineering well-structured software, tailored to the specific user needs and the application scenario. This methodology increases several software quality attributes like adaptability, modularity, traceability, readability, maintainability, extendibility, understandability reusability, flexibility, and ease of evolution.

IV. FEATURES AND FEATURE MODEL

"A feature is a structure that extends and modifies the structure of a given program in order to satisfy a stake holder's requirement, to implement and encapsulate a design decision, and to offer a configuration option".

It is a unit of functionality which satisfies a requirement, represents a design decision, and provides a potential configuration option [5]. They are modular entities which encapsulate a particular functionality of the system.

They help in the explanation of commonality and variability during the analysis, design, and implementation phases of software product lines. Classification of features increases the comprehensibility of a product line. By selecting and removing features, software provides different facilities and different configurations. Typically, from a collection of features, many different software systems can be generated that share some common features and at the same time differ in others. A feature model diagram represents all the products of the software product line [6].

Hierarchically arranged features of a feature model can be classified as [5]:

a) Mandatory

b) Alternative feature group

c) Optional

d) Or feature group

e) Excluded

f) Includes

Fig. 1 shows a sample feature model prepared by the Eclipse FeatureIDE plugin. This feature model is modeling a modified SPL of famous *hello world* program. The features GoodMorning and World are mandatory and simply print the features name. The features Pretty and Beautiful are alternatives, but not required. This SPL feature model contains three valid configurations of *hello world* programs.

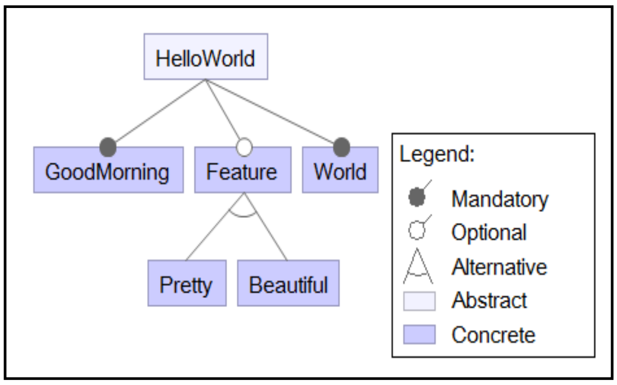

Fig. 1. Sample feature model for *hello world* program.

V. SOFTWARE QUALITY ATTRIBUTES

ISO defines the term quality as –"the totality of characteristics of an entity that bear on its ability to satisfy stated or implied needs".

For any product line to continue to function successfully and evolve as per need, it is imperative to look upon all the quality attributes that may affect it in future. Same is the case with feature models which are an inseparable part of SPLE.

Quality attributes are categorized into two types: internal and external. Internal quality attributes are directly measured on the basis of product features such as size, length, or complexity. Whereas external quality attributes, e.g. efficiency, reliability, usability and maintainability etc can only be measured with respect to how a software system relates with its environment and therefore, are measured once the software system is fully developed and deployed. Since external quality attributes are hard to evaluate in early phases of the software development process, indirect measurement based on internal quality attributes is often devised. The reason being, that internal quality attributes are appropriate determinants for external quality attributes [7].

Usability quality attribute can be defined in terms of ease of use, i.e. models should be user friendly. In other words, they should be easy to learn, navigation should be simple, easy to use for input preparation, operation, and interpretation of output, easy for new or infrequent users to learn or use.

It defines how well the feature model meets the requirements of the user i.e. the variability and commonality. It is concerned with evaluating how well the model is understandable and communicative. It also affects reusability of feature models, which is a good cost efficient and time saving development way. Feature models should be generic enough to be used easily across different application. Usability is a design quality and it defines the capability for feature models to be suitable for use in other applications and in other scenarios. It minimizes the duplication of features and also the implementation time [8].

SPL feature models should be communicative. They should provide useful inputs and outputs that can be assimilated. A less communicative feature model will not solve the purpose of usability. Rather than using its internal terminology, it should use the users' terminology to communicate with them. It should make use of users' background knowledge.

Usability is also affected by understandability. The more the understandability the lesser the effort needed to recognize the logical concept behind the feature model. This quality attribute is very important as the design of the feature model will affect the understandability of the whole product line. Size and complexity are important measures here as they directly affect understandability. Increased understandability leads to better management of the software product line. Wrong interpretations of feature models can lead to misunderstood and faulty product lines. It is difficult to manage and improve the product line without understanding the whole process. Therefore understandability of feature models has a lot of influence on software product lines and affects the quality of the same.

Learnability defines how easy the model is to learn, in the sense that users can start using it quickly. In reference to feature models it means that the operations of the model should be easy to understand and learn, while being observed. This attribute assures satisfying use of the feature model. It contributes to usability, because usability is defined as “easy to learn”.

Our focus is on measuring communicativeness, understandability because this will allow users to reach a reasonable level of usage proficiency that too within a short time and hence improved re-usage. It will eventually help in achieving specified performance and optimal performance.

The complexity of a model is based on the number of (different types of) features and on the number of (different types of) (dynamically changing) relationships (or interactions) between them. In SPL feature models, when features are added, the variability is increased. This in turn increases the complexity of the feature model. It is seen that high complexity results in reduced understandability which impedes the analyzability, adaptability and flexibility of the model, amongst other model qualities. This relationship between structural complexity and external quality properties like understandability and modifiability has been repeatedly demonstrated in various works. Feature models which are less communicative and understandable are difficult to analyze, modify, extend, integrate, and also reuse. To achieve the promised benefits of SPL in terms of increased reusability and productivity, it is necessary to control understandability. Understandability, per se, is not an easy-to-measure quality attribute in the early stages of the software development process.

VI. METRICS

The assessment of feature models can be done with the help of metrics. Metrics are software measurement units, which measure the degree to which a given system, component or process possesses a given attribute [9]. Metrics use numerical ratings to measure various domains like the complexity and reliability of source code, the length and quality of the development process and the performance of the application when completed. These metrics also serve to improve the quality of the resulting software products by helping to predict the possible quality of the final system and improve the product line based on these predictions [10].

Despite the emergence of methods and techniques, the need of measures for assessing quality attributes in software product line feature model still needs to be fulfilled. Study shows that very limited work has been done in the field of defining and validating metrics and assessment of the internal and external quality attributes in reference to feature models. Oliveira et. Al , Asim Rahman, Zhang et. Al. have proposed metrics for accessing quality of product line architecture. Assessment of the architectural quality is important but it is equally important to design methodologies and processes for creating high quality software product line conceptual models, which are most often in the form of SPL feature models. But no well-established and acknowledged methodology is available. The lack of appropriate mechanisms for measuring the properties of software product lines can be a reason for this. A set of structural metrics have been proposed by Bagheri et.al for assessing the maintainability of software product lines feature model. But the core focus of software product lines is on reusability and the metrics proposed in the paper were found to be inefficient to assess the same. The author has used measures for SPL feature models proposed by Briand et.al. TABLE I contains few measures [11].

TABLE I. MEASURES FOR SPL FEATURE MODEL

MEASURE TYPE	MEASURE NAME
SIZE MEASURE	NUMBER OF FEATURES (NF)

	NUMBER OF TOP FEATURES (NTOP)
	NUMBER OF LEAF FEATURES (NLEAF)
STRUCTURAL COMPLEXITY MEASURES	CYCLOMATIC COMPLEXITY (CC)
	CROSS TREE CONSTRAINTS (CTC)
	RATIO OF VARIABILITY (ROV)
	COEFFICIENT OF CONNECTIVITY-DENSITY (COC)
	FLEXIBILITY OF CONFIGURATION (FOC)
LENGTH MEASURE	DEPTH OF TREE (DT)

These measures are useful to assess the quality of feature models, but do not suffice in analyzing the usability of the whole system.

These measures need to be further studied by employing classical statistical correlation techniques in order to understand how well each of the structural metrics can serve as discriminatory references for attributes like usability and it sub characteristics like communicativeness and understandability. In other words the efficiency of these metrics for supporting the external quality attribute prediction needs to be analyzed. A set of structural measures is needed to identify the degree of understandability and communicativeness of the feature model.

VII. CONCLUSION

Many software engineering researchers have proved that measurement is a good means of improving software quality. But only handful researchers have addressed the issue of proposing appropriate structural quality metrics for software product line feature models. Available generic metrics need to be analyzed and applied to assess the quality of the software product lines feature model. The requirement of valid metrics and the limitations of the currently available metrics, should give motivation to researchers to work in this direction. These measures will act as early indicators of the perceived subjective value of usability, communicativeness and understandability. Analogous to metric design for other software engineering discipline, proposed measures are not comprehensive and other advanced research will further complete this proposed set by defining new metrics from other perspectives as well.

REFERENCES

[1] P. Clements and L. Northrop, Software Product Lines: Practices and Patterns, Addison-Wesley, Boston, 2001.
[2] M. Svahnberg, J.Bosch, “Evolution in Software Product Lines”, Journal of Software Maintenance: Research and Practice, November/December 1999, Volume 11, Issue 6, Pages 391–422.
[3] S. Apel and C. Kastner, “An Overview of Feature-Oriented Software Development,” J. Object Technology (JOT), Volume 8, No. 5, 2009, Pages 49-84.
[4] A. Sharma, S.S. Sarangdevot, “Investigating the Application of Feature-Oriented Programming in the Development of Banking Software Using Eclipse-Featureide Environment”, International Journal of Computer Science & Technology, March 2011,Volume 2, Issue 1 ,Pages 53-57.
[5] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study", Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report Cmu/Sei-90-Tr-021 , 1990.
[6] D. B. Cuevas, “On The Automated Analysis of Software Product Lines using Feature Models-A Framework for Developing Automated Tool Support”, The Department of Computer Languages and Systems, Spain, June 2007
[7] M. Barbacci, M. Klein, T. Longstaff, and C.Weinstock, “Quality attributes,” SEI, December,1995.
[8] H. Al-Kilidar, K. Cox, and B. Kitchenham, “The use and usefulness of the ISO/IEC 9126 quality standard,” in 2005 International Symposium on Empirical Software Engineering,2005. IEEE, 2005, p. 7.
[9] N. Fenton, “Software measurement: A necessary scientific basis,” IEEE Transactions on software engineering, pp. 199–206, 1994.
[10] L. Briand, J. Wust, J. Daly, and D. Victor Porter, “Exploring the relationships between design measures and software quality in object-oriented systems,” Journal of Systems andSoftware, vol. 51, no. 3, pp. 245–273, 2000.
E. Bagheri, D. Gasevic, “Assessing the Maintainability of Software Product Line Feature Models Using Structural Metrics”, Springer, September 2011, Volume 19, Issue 3, Pages 579-612.