International Journal of Information and Computation Technology.
ISSN 0974-2239 Volume 3, Number 4 (2013), pp. 355-360
© International Research Publications House
http://www. irphouse.com /ijict.htm

Software Product Line Engineering: Paradigm for Product Family

Geetika Vyas[1], Amita Sharma[2] and Astha Pareek[3]

[1]Dept of CS & IT, The IIS University, Jaipur, Rajasthan, INDIA.
[2,3]Dept of CS & IT, The IIS University, Jaipur, Rajasthan, INDIA.

Abstract

The modeling foundation of Software Product Line Engineering (SPLE) is the segregation of variant features of all the products which belong to a family. In brief, its aim is to catalog what is common and what differs between products. Feature is a basic term associated with SPL and feature model diagram portrays the product deviation. SPLE is a powerful approach to increase the efficiency of the software engineering process and variety of software system can be developed from a single software product line. Therefore it should be realized that a low quality design can ripple through many generated software systems. This state of the art presents a comprehensive overview of the software product lines theory and discusses the relationship between feature models and the various qualities attributes affecting them .The available methods (metrics) to measure these attributes are also discussed in brief.

Keywords: software product line, feature, feature model, quality, metrics.

1. Introduction

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 possible 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 solution to these not so new, but increasingly challenging problems.

SPLE is an approach that develops and maintains families of products taking advantage of their common aspects and predicted variability's at the same time. SPLE focuses on reuse in systems development, as a viable and important software development paradigm.

The focus of this paper is on and around SPLE and feature models. Section II contains the basic concepts related to this. Section III focuses on quality attributes of feature models. Section IV discusses the various quality attributes measures and scope for future work. Section V contains references.

2. Software Product Line

As defined by Clements, Software Product Line is "A set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment".

SPLE includes software engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production. It focuses on systematic reuse instead of ad hoc reuse, leading to business benefits like improved time to market, enhanced performance, reduced development and maintenance costs, mass customization and improved software quality. In SPLE, a line means a set of software products that are related and share commonalities like data structures, software components and architecture etc.

This approach is structured into two main processes: Domain Engineering and Application Engineering. Domain engineering finds, records, constructs and presents set of software artifacts that can be used in future software's who specialize in that particular application domain. As a result, domain knowledge is achieved in the form of reusable software assets, which leads to faster software production and reduced time-to-market. It is the basis of SPLE and is also called engineering-for-reuse whereas application engineering which produces the individual applications is often called engineering-with-reuse. It centers on development with reuse, and develops the final products, using the core assets and the specific requirements expressed by the customers.

3. Features and Feature Models

"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 that satisfies a requirement, represents a design decision, and provides a potential configuration option. They are modular entities encapsulating a particular functionality of the system.

Features are helpful in the explanation of commonality and variability in the analysis, design, and implementation of software product lines. With the selection and removal of features, software provides different facilities and different configurations.

Typically, from a set 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 represents all the products of the software product line. These feature models are visually represented by feature diagrams. Hierarchically arranged features of a feature model can be classified as:

- Mandatory
- Or feature group
- Optional
- Excluded
- Includes
- Alternative

Figure 1 shows a sample feature model of E-Shop. It has 3 mandatory features viz. catalogue, payment, security and 1 optional feature viz. search. The mandatory feature payment has two options viz. bank transfer or credit card. The security feature has two alternative features viz. high or standard.

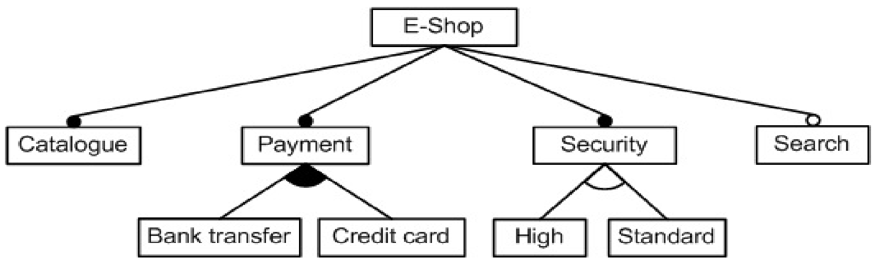

Figure 1: Sample feature model of E-Shop.

4. Software Quality Attributes

According to ISO the term quality can be defined 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 and evolve as needed, it is imperative to look upon all the quality attributes that may affect them in future. Same is the case with feature models which are an inseparable part of SPLE.

Quality attributes can be categorized into two types: internal and external. Internal quality attributes can be directly measured on the basis of product features such as size, length, or complexity. Whereas external quality attributes, e.g. efficiency, reliability and maintainability 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, an indirect measurement based on internal quality attributes is devised. The reason being, that internal quality attributes are appropriate determinants for external quality attributes.

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 application 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 application. Out of these usability and complexity play special role in SPL and feature models.

Usability which is a user quality 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. Reusability 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.

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 complex are difficult to analyze, modify, extend, integrate, and 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. Therefore, indirect measures based on the structural complexity of the model are often useful.

The assessment of complexity 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. 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.

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 1: Measures for SPL feature models.

Measure type	Measure name
Size measure	Number of features (NF)
	Number of top features (NTop)
	Number of leaf features (NLeaf)
Length measure	Depth of tree (DT)
Structural complexity measures	Cyclomatic complexity (CC)
	Cross tree constraints (CTC)
	Ratio of variability (RoV)
	Coefficient of connectivity-density (CoC)
	Flexibility of configuration (FoC)

From the above table, various numeric values can be generated for any feature model. For the E-Shop feature model we can derive various values like NF=9,NTop=4,NLeaf=6,DT=2,RoV=1,FoC=0.11 and hence forth. These values can 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. But it should be noted that analogous to metric design for other software engineering discipline, this set of metrics is not be comprehensive and other advanced research can further complete this proposed set by defining new metrics from other perspectives as well.

5. 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.

References

[1] Asim Rahman (2004), "Metrics for the Structural Assessment of Product Line Architecture", Master Thesis, School of Engineering Blekinge Institute of Technology, Sweden, Thesis no: MSE-2004:24.

[2] Ian Sommerville (2008), *Software Engineering*, Pearson Education, India.

[3] E. Bagheri, D. Gasevic (2011), "Assessing the Maintainability of Software Product Line Feature Models Using Structural Metrics", *Springer*, Volume 19, Issue 3, 579-612.

[4] D. M. Weiss, P. C. Clements, K. Kang, and C. Krueger (2006), " Software product line hall of fame, " *in SPLC '06:Proceedings of the 10th International on Software Product Line Conference*, Washington, DC, USA: IEEE Computer Society, 237.

[5] E. A. Oliveira Junior, J.C. Maldonado, I.M. S. Gimenes (2010), "Empirical Validation of Complexity and Extensibility Metrics for Software Product Line Architectures", *Proceedings of the 2010 Fourth Brazilian Symposium on Software Components, Architectures and Reuse*, 31-40.

[6] Manuel F. Bertoa, Jose´ M. Troya, Antonio Vallecillo (2006), "Measuring the usability of software components", *The Journal of Systems and Software*, 427–439.

[7] T. Zhang, L. Deng, J. Wu, Q. Zhou, and C. Ma (2008), "Some Metrics for Accessing Quality of Product Line Architecture", *Proceedings of the 2008 International Conference on Computer Science and Software Engineering*, Washington, DC, USA, IEEE Computer Society, (500–503).