

Available online at <http://www.mecs-press.net/ijeme>

# Analysis of Features using Feature Model in Software Product Line: A Case Study

Hitesh Yadav<sup>a\*</sup>, A. Charan Kumari<sup>b</sup>

<sup>a</sup>Assistant Professor, The NorthCap University, Gurugram, India

<sup>b</sup>Associate Professor, The NorthCap University, Gurugram, India

Received: 13 February 2017; Accepted: 11 September 2017; Published: 08 March 2018

---

## Abstract

This paper shows an analysis of features of email system using feature model in a Software Product Line (SPL). The core features that can be used by different SPLs are identified using feature model. The analysis is based on two primary measures – reusability and consistency. Reusability measures the level of frequency of usage of the feature in developing a new software product line and consistency ensures that the core features are consistent in a software product line. On the basis of reusability measure, the core features are classified into four different categories. These measures help in understanding the Return on Investment in a software product line.

**Index Terms:** Feature Model, Core asset, Software Product Line, Feature.

© 2018 Published by MECS Publisher. Selection and/or peer review under responsibility of the Research Association of Modern Education and Computer Science.

---

## 1. Introduction

Software Product Line (SPL) engineering is a way to build the configurable software product. Software Product Line helps to record the variabilities and commonalities. The main challenge is to obtain a new software product line configuration which helps to minimize the cost and resource requirement. Software product line is built by using a set of components that allow variability. Software product line can be developed in two ways a) domain engineering and b) product derivation. The process of creating the common and variable software feature of the SPL is called domain engineering. The process of changing the reusable software features according to the requirements of customers is called product derivation.[15] In the case of a proactive approach, all features of SPL are constructed after the domain engineering. But in extractive approach existing features a new software product line is constructed.

\* Corresponding author.

E-mail address: [hiteshi.yadav@gmail.com](mailto:hiteshi.yadav@gmail.com)

SPL helps an organization to develop their products by using reusable features rather than starting from scratch. This core function of the Software Product Line decrease functionality cost of software system [15]. SPL improves the quality of product and control production cost and ensures discipline in software.

Software product line is a planned process rather than anticipated. One of the examples of SPL is e-commerce organization. Every e-commerce company has many channels for selling their product to the customer. Some of the components are common to the channels and some of them are different.

This paper has been organized as follows. Section 2 shows related work. The feature model of e-mail system and its measurement and analysis is described in sections 3 and 4 respectively. Section 5 has the conclusion.

## 2. Related Work

Software Product Line is prominent concept since the eighties and now it has gained the attention of the researchers and developers. Since the market is evolutionary, so it is required to replace software product once they prevent the goals of the organization. Some of the researchers have focused on existing approaches, methods and tools for product lines. Some have proposed their method for effective product lines, which can accommodate changes.

Software product line is defined in [13][17] as a software-intensive system. Author has explained a framework for product line practice. A framework helps to identify concepts underlying SPL and necessary activities before creating a product line. The development of software product line involves a development of core assets and development of products by using reusable core assets. Using existing products can develop new products. This provides immense benefits to the organization.

In [20] author introduced Aspect-Oriented Programming (AOP), which improved assembling process in SPL. Author find existing method is not effective and for the implementation of software product line needs high cost. The author used commonalities and variability are in the assembly process and used AOP concept to analyze requirements and design.

A systematic method is introduced by [11] which is called variation point model. It is used for developing components for reusability in many applications. The paper described SPL as a reusability approach and various different methods to model variability i.e. using information hiding, parameterization, inheritance, and variations points. Software Product Line Integrated technology (SPLIT), PulsE, KobrA [18], and Wheels [19] helps in building product lines.

A product line evolution method is proposed in [2] based on kaizen approach, which is used in Japanese industry. This approach defines some work standard that continues improve processes in a product line. The author defined various types of matrices in which core assets can be measured and on the basis of that author further classified core assets. Several measurement definitions are defined in [1] in which it defines the metric index and analyzed the relationship of core asset. Author has given three measurement index i.e. consistency, reusability, and coverage in which it analyzes the relationship with the management of core asset.

In [3] author described various matrices to measure core assets, which helped in correlating the evolution of core asset library. Author defined three measures namely reusability, coverage and consistency. On the basis of evolvment coefficient author classified core assets library levels into a form of optimized and stabilized form.

In [21] Author took a case study for evaluation of complexity, modularity, and stability of core asset. The author presented a decision model and performed a quantitative study for implementing services as software product line core asset.

A framework called Wheels is presented [19] for software product line development. Wheels framework is a component of Software Product Line Integrated technology (SPLIT) method. The author tried to investigate SPLIT under which Wheels method works. Wheels is a tailorable process which offers the ability to adapt to different situations.

A survey based on search-based software engineering (SBSE) for software product lines was done by M. Harman in the paper [4]. Software product line is a group of related product, they share few common feature but differ in some specific feature. These differences in the features help product line to find variability in the

system.





In [22] [23], Author identified various dimensions and extended notations of business process modeling notations for handling variability. By using a set of operations, a process model can be individualized according to the organization requirements.

Xinyu *et al.* 2011 define the design and realization of core asset and presents system architecture of core asset library [6]. Mikyeong *et al.*, 2009 focus on reuse to build high quality and cost effective system. [7] [9]

### 3. Feature model of E-mail System

Feature models are important to understanding the variability of software systems. This model tells that which features can be used with other to form a product and which ones cannot. So, feature model is used to combine features for software product line. This model looks like a tree structure. Every product obtained from feature model represents a unique set of features. Fig. 1 shows the feature model of the e-mail system. Each feature has one parent except root node and has set of children. The symbols used to represent different relationships in a feature model are shown in Table 1.

Table 1. Relationship Symbols in Feature Model

Symbols	Relationship
	Mandatory Feature
	Optional Feature
	Inclusive Relationship
	Exclusive relationship

Feature model has four types of relationship - First is a Mandatory relationship in which a feature is selected when parent feature is selected. For example, *Utility* and *Message Compose*. Second is an Optional relationship in which a feature may or may not be selected if the parent feature is selected. For example, a feature *Drive*. The third is an Exclusive relationship in which exactly one feature is selected among others whenever parent feature is selected. For example *Docs*, *Sheets* and *Slide*. Fourth is an inclusive relationship in which at least one feature must be selected if parent feature is selected. For example, *Voice Talk*, *Chat* and *Video Talk* [5]. A feature model is designed using AND-OR graph.

#### i. Software Product Line of E-mail System

A Product matrix displays a list of features or core asset for the email system product line. A Product matrix as represented in Fig. 2 is a two- dimensional matrix that shows the relationship between the features and members of a product line. These features can be shared by different systems in the product line. Tick mark represents features accommodated by different product line. In the first column of the table, we write all the features and others columns represent different product lines. From the product matrix, different product lines can be derived. For example, product line A represents communication between two users in three different modes – voice talk, chat and video.

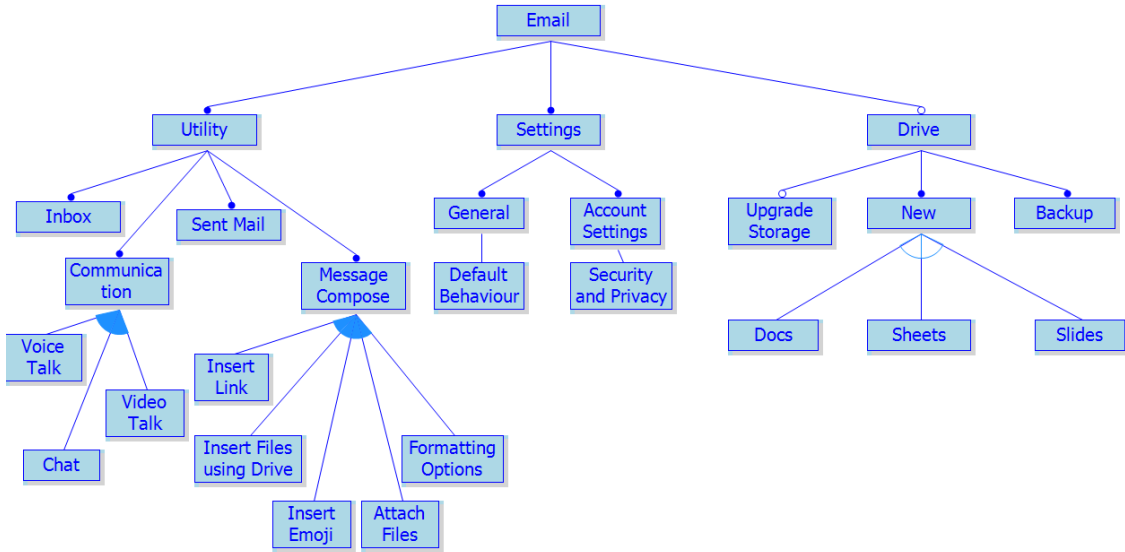


Fig.1. A Simplified Feature Model of E-mail System Product Line

#### 4. Feature Measurement & Analysis

Reusability and consistency are two important measures to understand the nature of core features and to estimate the return on investment for an organization. Jianjie *et al.*, 2011 proposed the measurements for the reusability and consistency of the features.

##### i. Reusability

Reusability [1] plays a big role in the measurement of a core feature. It measures the level of frequency of usage of a feature in developing a new product line. In general, a reusable matrix will change dynamically due to the development of different product lines from time to time. And also, sometimes, it can be a sparse matrix too.

Let the reusable product matrix  $U = (u_{ij})_{n \times m}$ ; with ‘n’ features and ‘m’ product lines.

$u_{ij} = 1$ , if the feature  $r_i$  is used in the product line  $P_j$  and 0 otherwise.

**Measurement 1 (frequency of usage of a feature):** The frequency of usage of a specific feature can be calculated using the formula [1]

$$u_i = \sum_{j=1}^m u_{ij} \tag{1}$$

For the above case study the reusability of different features is shown in Table 2. In first row of table 2, frequency of usage of email is 11 and also it is evident from the Fig. 2, that the email feature is being used in every product line. It is clearly seen that *utilities* feature is using in four product lines so the frequency of usage is 4.

Features	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9	P 10
Email	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Utilities	✓	✓	✓								✓
Communication	✓	✓	✓								✓
Voice	✓										✓
Chat		✓	✓								
Video Talk			✓								
Message Compose				✓	✓						✓
Insert files using drive				✓							
Insert emoji		✓			✓						
Attach files					✓						
Formatting Options					✓						
General						✓					
Default Behavior						✓					
Account Setting							✓				
Security and privacy							✓				
Drive								✓	✓		
New								✓	✓		
Docs								✓			
Sheets									✓		
Slides									✓		
Backup										✓	
Upgrade Storage										✓	

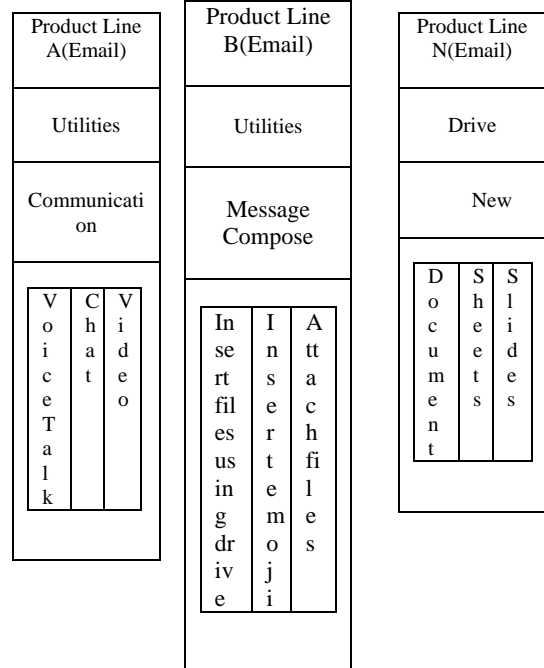


Fig.2. Different Product Lines Derived from the Product Matrix.

**Measurement 2 (reusability of a feature):** The reusability of a specific feature [1] is computed as specified in equation 2.

$$R_i = \frac{\sum_{j=1}^m u_{ij}}{m} \quad (2)$$

In other words, reusability can be calculated by dividing the frequency of usage of a feature with the total number of product lines. The calculated values for the e-mail case study are shown in Table 3.

In addition, we can also compute the maximum and minimum reusable feature as below.

Table 2. Frequency of usage of features in Email System.

Feature	Frequency of usage
Email	11
Utilities	4
Communication	4
Voice Talk	2
Chat	2
Video	1
Message Compose	3
Insert file using drive	1
Insert emoji	2
Attach files	1
Formatting options	1
General setting	1
Default behavior	1
Account setting	1
Security and privacy	1
Drive	2
New	2
Docs	1
Sheets	1
Slides	1
Backup	1
Upgrade storage	1

**Measurement 3:** The maximum reusable feature [1] is  $R_k = \max (R_1, R_2, R_3, \dots, R_{22})$ . Feature ID  $R_1$  has maximum value of 11 among others. So it is obvious from table 3 that  $R_1$  has maximum reusability.

**Measurement 4:** Minimum reusable feature [1],  $R_k = \min (R_1, R_2, R_3, \dots, R_{22})$  It is evident from table 3 that  $R_6, R_8, R_{10}, R_{11}, R_{12}, R_{13}, R_{14}, R_{15}, R_{18}, R_{19}, R_{20}, R_{21}, R_{22} = 1/11$

Based on the reusability measure the features are divided into four categories - Type 1, Type 2, Type 3 and Type 4 core features as represented in Table 4. This classification helps in understanding the most reusable feature and the least reused ones. According to the table 4, Type 1 has a single feature of feature ID  $R_1$  (feature: *email*). Features like *utilities*, *communication*, *compose message* lies in Type 2. *Voice*, *Chat*, *Video*, *Insert file using drive*, *Features like Insert emoji*, *Attach files*, *Formatting options*, *General setting*, *Default behaviour*, *Account setting*, *Security and privacy*, *Drive*, *New*, *Docs*, *Sheets*, *Slides*, *Backup*, *Upgrade storage* lies in Type 3. No feature lies in Type 4.

Table 3. Reusability of features in Email System.

Feature Name	Feature ID	Reusability
Email	R <sub>1</sub>	1
Utilities	R <sub>2</sub>	0.36
Communication	R <sub>3</sub>	0.36
Voice	R <sub>4</sub>	0.18
Chat	R <sub>5</sub>	0.18
Video Talk	R <sub>6</sub>	0.05
Message Compose	R <sub>7</sub>	0.27
Insert file using drive	R <sub>8</sub>	0.05
Insert emoji	R <sub>9</sub>	0.18
Attach files	R <sub>10</sub>	0.05
Formatting options	R <sub>11</sub>	0.05
General setting	R <sub>12</sub>	0.05
Default behavior	R <sub>13</sub>	0.05
Account setting	R <sub>14</sub>	0.05
Security and privacy	R <sub>15</sub>	0.05
Drive	R <sub>16</sub>	0.18
New	R <sub>17</sub>	0.18
Docs	R <sub>18</sub>	0.05
Sheets	R <sub>19</sub>	0.05
Slides	R <sub>20</sub>	0.05
Backup	R <sub>21</sub>	0.05
Upgrade storage	R <sub>22</sub>	0.05

## ii. Consistency

Consistency is a qualitative measure that indicates how consistent a core feature is in a product line. Therefore core features do not contradict each other. Consistency ensures that the core features are consistent in a product line. [1]

**Measurement 5 (of any two core features):** The consistency of two core features can be computed as shown in equation 3 [1].

$$Consistency(r_i, r_j) = \frac{\sum_{k=1}^m u_{ik} u_{jk}}{m} \quad (3)$$

**Measurement 6 (of k number of core features):** the consistency of a set of core features can be computed as shown in equation 4.

$$Consistency(r_i, r_j, \dots, r_k) = \frac{\sum_{k=1}^m u_{ik} u_{jk} \dots u_{nk}}{m} \quad (4)$$

For example, the consistency of some of the pairs of features is listed in Table 5. Which are evaluated as follows:

By using the formula 4, the consistency of *Email, Utilities* is 0.36. In a similar way, the consistency of *communication and voice talk* features is 0.18 and the consistency of *slides, backup* is 0.

Table 4. Classification of Features

Type	Range	Feature
Type 1	$R \geq 0.5$	R <sub>1</sub>
Type 2	$0.2 \leq R < 0.5$	R <sub>2</sub> , R <sub>3</sub> , R <sub>7</sub>
Type 3	$0.05 \leq R < 0.2$	R <sub>4</sub> , R <sub>5</sub> , R <sub>6</sub> , R <sub>8</sub> , R <sub>9</sub> , R <sub>10</sub> , R <sub>11</sub> , R <sub>12</sub> , R <sub>13</sub> , R <sub>14</sub> , R <sub>15</sub> , R <sub>16</sub> , R <sub>17</sub> ,
Type 4	$R < 0.05$	R <sub>18</sub> , R <sub>19</sub> , R <sub>20</sub> , R <sub>21</sub> , R <sub>22</sub> No feature

## 5. Conclusion

This paper presented the analysis of features using a feature model in an e-mail software product line. The motive is to analyze the core features in a SPL in terms of the two vital attributes – reusability and consistency. This helps in understanding the most reusable and consistent core features and consequently assists in deriving new software product lines. In the e-mail system, it has been found that some core features are highly reused and some to a lesser extent. This understanding helps organizations in developing new SPLs with a higher return on investments.

Table 5. Consistency of Features

Features	Consistency
Email, Utilities	0.36
Communication, Voice Talk	0.18
Slides, Backup	0
Backup, upgrade	0.09
Docs, Sheet	0
New, Docs	0.09
Drive, New	0.18
Insert file using drive, emoji	0
Emoji, attach file	0.09
Attach file, Formatting options	0.09
Email, Utilities, Communication	0.36
Utilities, Communication, Voice Talk	0.18

## References

- [1] Jianjie Ding, Kegang Hao and Hong Hou. The research on measurement and management of core asset library, 2011, 2nd International Conference on Artificial Intelligence, Management Science and Electronic



- Commerce (AIMSEC), Deng Leng, 2011, pp. 3542-3545.
- [2] Mari I, Yoshiaki F. Software Product Line Evolution Method Based on Kaizen Approach, SAC' 07, March 11-15, 2007.
  - [3] D. jianjie, HaoKeGang, HouHong, GuoXiaoQun, Metric-based evolution analysis and evaluation of software core assets library, IEEE 2009, page no. 799-803.
  - [4] M Harman, Y Jia, J Krinke, WB Langdon, J Petke & Y. Zhang. Search based software engineering for software product line engineering: a survey and directions for future work Proceedings of the 18th International Software Product. 2014
  - [5] Lukas Linsbauer, Roberto Erick Lopez-Herrejon, Alexander Egyed. Feature model synthesis with Genetic Algorithm, Search-Based Software Engineering: 6th International Symposium, SSBSE 2014, Fortaleza, Brazil, August 26-29, 2014, page no 153-167.
  - [6] X. Zhang, L. Zheng and Y. Lu. The Exploration of the Core Asset Library in the Software Product Line, 2011 Eighth International Conference on Information Technology: New Generations, Las Vegas, NV, 2011, pp. 1082-1083.
  - [7] Mikyeong Moon and Keunhyuk Yeom. An approach to developing core assets in product line, 11th Asia-Pacific Software Engineering Conference, 2004, pp. 586-588.
  - [8] C. Frangois, C. Roger. A Product Line engineering practices model, Science of Computer Programming, vol. 57, pp. 73, 2005.
  - [9] Moon, Mikyeong and Yeom, Keunhyuk. An Approach to Develop Requirement as a Core Asset in Product Line, Software Reuse: Methods, Techniques, and Tools: 8th International Conference, ICSR 2004, Madrid, Spain, July 5-9, 2004. Proceedings, 2004, pp 23-34.
  - [10] Joaquin Peña, Michael G. Hinchey, Manuel Resinas, Roy Sterritt, James L. Rash. Designing and managing evolving systems using a MAS product line approach, Science of Computer Programming, Volume 66, Issue 1, 15 April 2007, pp 71-86.
  - [11] Diana L. Webbera, Hassan Gomaab. Modeling variability in software product lines with the variation point model, ELSEVIER, April 2003.
  - [12] Felix Bachmann, Paul C. Clements. Variability in Software Product Lines, Software Engineering Institute, September 2005.
  - [13] P. Clements, L. Northrop, Software Product Lines Practice and Patterns, Addison-Wesley, 2002
  - [14] Matthias Galster, Danny Weyns, Dan Tofan, Bartosz Michalik, and Paris Avgeriou, Variability in Software Systems— A Systematic Literature Review, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 40, NO. 3, MARCH 2014, pp 282- 306.
  - [15] Jianmei Guo, Jules White, Guangxin Wang, Jian Li, Yinglin Wang. A genetic algorithm for optimized feature selection with resource constraints in software product lines, Journal of Systems and Software, Volume 84, Issue 12, December 2011, page no. 2208-2221.
  - [16] Brownsword, Lisa; & Clements, Paul. A Case Study in Successful Product Line Development. Software Engineering Institute, Carnegie Mellon University. 1996.
  - [17] Clements, Paul a Framework for Software Product Line Practice SEI Technical Report, 1992.
  - [18] Joachim Bayer, Dirk Muthig, and Brigitte Goepfert. The Library Systems Product Line: A Kobra Case Study. Technical Report IESE-Report No. 024.01/E, Fraunhofer Institute for Experimental Software Engineering (IESE), November 2001.
  - [19] Michel Coriat, Frederic Waeber, Product line process framework: The Wheels Process.
  - [20] Seung-hyun Heo and Eun Man Choi. Representation of Variability in Software Product Line Using Aspect-Oriented Programming Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06), Seattle, WA, 2006, page no. 66-73.
  - [21] H. B. G. Ribeiro *et al.* An Assessment on Technologies for Implementing Core Assets in Service-Oriented Product Lines, 2010 Fourth Brazilian Symposium on Software Components, Architectures and Reuse, Bahia, 2010, pp. 90-99.
  - [22] D. K. Sharma, Hitesh and V. Rao, Configurable Business Process Modeling Notation, 2014 IEEE

International Advance Computing Conference (IACC), Gurgaon, 2014, pp. 1424-1429.

- [23] D. K. Sharma, Hitesh and V. Rao, Individualization of process model from configurable process model constructed in C-BPMN, International Conference on Computing, Communication & Automation, Noida, 2015, pp. 750-754.

### Authors' Profiles



**Hitesh Yadav** is pursuing PhD from The NorthCap University and is M.Tech. holder in Computer Science from Maharishi Dayanand University, Rohtak, India in 2012. Her current research interests include Software Engineering. Hitesh Yadav received the B.Tech degree in Information Technology with Honors from Maharishi Dayanand University, Rohtak in 2010.



**Dr. A. Charan Kumari** received her Ph.D from Dayalbagh Educational Institute, in collaboration with Indian Institute of Technology, Delhi, India, under their MOU. She has an excellent teaching experience of 17 years in various esteemed institutions and received various accolades including the best teacher award. Her current research interests include Search Based Software Engineering, Evolutionary computation and soft computing techniques. She has published papers in journals and conferences of national and international repute. Dr. Charan has also served as reviewer of various journals and conferences. She has delivered an invited talk at 43rd CREST open workshop on Hyper-heuristics at University College London, London. She is a member of IEEE, Computer Society of India (CSI) and Systems Society of India (SSI).

**How to cite this paper:** Hitesh Yadav, A. Charan Kumari, "Analysis of Features using Feature Model in Software Product Line: A Case Study", International Journal of Education and Management Engineering(IJEME), Vol.8, No.2, pp.48-57, 2018.DOI: 10.5815/ijeme.2018.02.06