*Article*

# Towards the maturity model for feature oriented domain analysis

**Muhammad Javed**[1], **Muhammad Naeem**[1], **Hafiz Abdul Wahab**[2]
[1]Department of Information Technology, Hazara University, Mansehra, Pakistan
[2]Department of Mathematics, Hazara University, Mansehra, Pakistan
E-mail: mjavedgohar@hotmail.com, naeem@hu.edu.pk, wahabmaths@yahoo.com

*Received 5 April 2014*; *Accepted 10 May 2014*; *Published online 1 September 2014*


**Abstract**
Assessing the quality of a model has always been a challenge for researchers in academia and industry. The quality of a feature model is a prime factor because it is used in the development of products. A degraded feature model leads the development of low quality products. Few efforts have been made on improving the quality of feature models. This paper is an effort to present our ongoing work i.e. development of FODA (Feature Oriented Domain Analysis) maturity model which will help to evaluate the quality of a given feature model. In this paper, we provide the quality levels along with their descriptions. The proposed model consists of four levels starting from level 0 to level 3. Design of each level is based on the severity of errors, whereas severity of errors decreases from level 0 to level 3. We elaborate each level with the help of examples. We borrowed all examples from the material published by the research community of Software Product Lines (SPL) for the application of our framework.

**Keywords** quality; feature models; maturity model; errors; dead features; invalid feature model.

**Computational Ecology and Software**
**ISSN 2220-721X**
**URL: http://www.iaees.org/publications/journals/ces/online-version.asp**
**RSS: http://www.iaees.org/publications/journals/ces/rss.xml**
**E-mail: ces@iaees.org**
**Editor-in-Chief: WenJun Zhang**
**Publisher: International Academy of Ecology and Environmental Sciences**

## 1 Introduction

Producing things in large amount require standardized processes, especially for the similar products. Companies are organizing their production in large amount of production (Benavides et al., 2010). To reuse existing systems in a systematic way, service-oriented systems resemble supply chain where products manufactured from supplied parts. Same case is for complex service-oriented systems, which needs third party services (Thomas, 2008). For example, car producer offer variation on a model with variable engines, gearboxes, audio and entertainment systems. Example of software services is online travel agency, which may use third-party services for hotel booking, invoicing and for payment option (Naeem, 2012). Similarly, increasing number of software systems with almost similar requirements guide us to Software Product Line (SPL) (Böckle and Linden, 2005). SPL Engineering helps in the development within application domain by considering their commonalities and variability. In SPL approach, products are being created by reusability

Bošković, 2011; Clements, 2001). SPL incorporating the property of similarities and variability in the family of softwares is a new technique in the development of softwares. This helps in the development of high quality software in a short period of time with low budget. Progress has been improved in the development by adopting SPL (Mendonça, 1999). Features represent the aspects of these software (Kang et al., 1990). To get a valid combination of these features we use feature model which depicts the relationships of these features and constraints on them (Batory, 2005). Usually feature models are tree like structures describing successive refinement of the variability in a product-line. Feature models were proposed back in 1990 as feature oriented domain analysis (FODA) (Kang et al., 1990).

The use of high quality process ensures the good quality resulting products. Hence, it is very important to investigate the quality of the selected model before putting it into practice. In other words, one can say that the quality of a feature model has prime importance because it contributes towards the development of high quality products. There are number of properties which affect the quality of a feature model. One of the agreed deficiencies in feature models is *errors* in the feature model.

There are some efforts in the literature on discovering errors in feature models, but to the best of our knowledge, no effort has been made on developing a framework based on those errors which should be able to comment on the quality of feature models (Ahmed, 2011; Batory, 2005; Batory, 2006; Benavides, 2007; Benavides, 2010; Thörn, 2007). So, there is a need of a technique which evaluates the feature model to represent the quality level of a feature model. This paper is a first attempt to present the framework to judge the quality of a given feature model, which we call *Maturity Model for FODA*.

The quality of a feature model can be analyzed from different perspectives which may includes: how efficiently it captures a given domain by keeping the integrity of model itself. The lesser are the occurrences of redundancies, anomalies and inconsistencies in a feature model, the more will be the integrity of a feature model (Maßen and Horst, 2004; Rosso, 2006). Although, in the case of feature models, number of quality improvement methods have been adopted, but there is still room to investigate the quality evaluation process (Thörn, 2007). The rest of the paper is arranged as follows: Section 2 provides the background information, whereas Section 3 presents the related work. Section 4 and 5 elaborates the feature model errors and explains FODA maturity model, respectively. The Section 7 concludes the paper.

## 2 Background

In this section, we provide the information which is important to understand the technical contribution of this paper.

Feature models were introduced by Kang in the form of technical report on FODA in 1990. A feature is prominent characteristic of a product (Kang et al., 1990). Feature model is a hierarchical model that captures the commonality and variability of SPL. The set of permissible selection of features from a feature model is called an instance (Rosso, 2006). Types of features allowed in feature models are discussed here.

*Mandatory feature:* If a feature is chosen then its mandatory feature must be selected in that instance (Benavides, 2010). It is represented by a filled circle at the end of edge. For example, *Call* is a mandatory feature of *Mobile Phone* in Fig. 1. This can be represented in the form of propositional formulas as $MobilePhone \leftrightarrow Calls$.

*Optional feature*: If a feature is selected in an instance then its optional sub-features can be selected or rejected depending on the preferences (Benavides, 2010). It is represented by empty circle at the end of edge. For example, *GPS* is an optional feature of *Mobile Phone* shown in Fig. 1. This can be represented by a propositional formula as $GPS \leftrightarrow MobilePhone$.

*Alternative-group:* A group of features having an alternative relevance with their parent means that exactly one

feature from this group must be selected if their parent is selected in an instance. It is represented by unfilled arc (Benavides, 2010). For example, features occurring under *Screen* make an Alternative-group in

The relationship in Fig. 1 can be represented in propositional logic as

$$\big((Screen \leftrightarrow Basic) \wedge \sim(Color \vee HighResolution)\big) \wedge \big((Screen \leftrightarrow Color) \wedge \sim(Basic \vee HighResolution)\big) \wedge \big((Screen \leftrightarrow HighResolution) \wedge \sim(Basic \vee Color)\big)$$

*Or-group:* For a group of features having an Or relevance with their parent means that at least one feature from this group must be selected, if their parent is selected in an instance (Rosso, 2006). An Or-group is shown by a filled arc. For example, features occurring under *Media* are making an Or-group in Figure 1. This can be captured by propositional formula as $\big(Media \leftrightarrow (Camera \vee MP3)\big)$

Apart from the parent child relationship, a feature diagram may have cross-tree constraints which are discussed below.

*Requires constraint:* If a source of requires constraint is selected that its target must also be chosen in that instance. This is represented by the dashed arrow that starts from the source and heads towards the target feature.

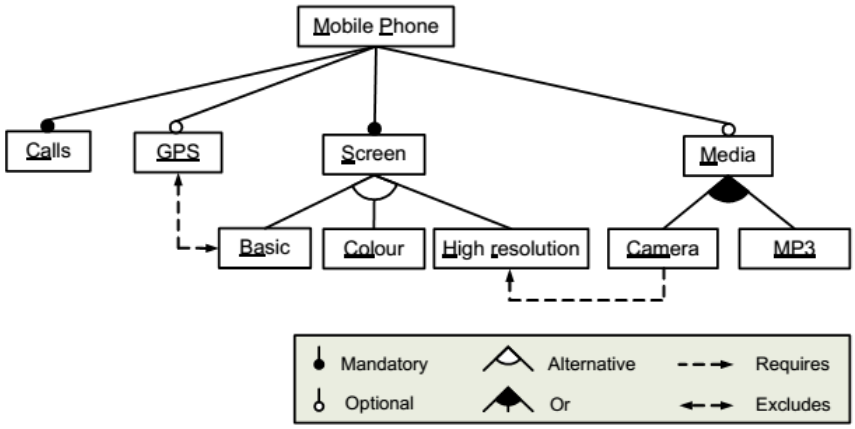


**Fig. 1** A feature model of a mobile phone (Benavides, 2010).

For example, requires constraint is shown between *Camera* and *High resolution* features in Fig. 1.

*Excludes Constraint:* The source and target features of excludes constraint cannot be selected in an instance.

This is represented by double headed dashed arrow, as shown between *Basic* and *GPS* features in Fig. 1.

Thus the feature diagram shown in Fig. 1 captures the following instances[1]:

$\{MP, Ca, Sc, Ba\}$, $\{MP, Ca, Sc, Col\}$, $\{MP, Ca, Sc, Hr\}$, $\{MP, Ca, GPS, Sc, Col\}$, $\{MP, Ca, GPS, Sc, Hr\}$, $\{MP, Ca, Sc, Hr, Me, Cam\}$, $\{MP, Ca, Sc, Ba, Me, MP3\}$, $\{MP, Ca, Sc, Col, Me, MP3\}$, $\{MP, Ca, Sc, Hr, Me, MP3\}$, $\{MP, Ca, Sc, Hr, Me, Cam, MP3\}$, $\{MP, Ca, GPS, Sc, Col, Me, MP3\}$, $\{MP, Ca, GPS, Sc, Hr, Me, MP3\}$, $\{MP, Ca, GPS, Sc, Hr, Me, Cam\}$, $\{MP, Ca, GPS, Sc, Hr, Me, Cam, MP3\}$

More formally: a feature diagram and its instance can be defined as:

**Definition 1**

[1]For brevity, we use underlined characters to represent features in the instances and further in the logical formulas

Feature Model and Instance – adapted from (Rubin and Chechik, 2012)

*Given a universe of elements* $\mathbb{F}$ *that represent features, a feature model* $\mathcal{FM} = \ll \mathcal{F}, \phi \gg$ *is a set of features* $\mathcal{F} \in 2^{\mathbb{F}}$ *and a propositional formula* $\phi$ *defined over the features from* $\mathcal{F}$*. An instance* $\mathcal{Ist}$ *of* $\mathcal{FM}$ *is a set of selected features from* $\mathcal{F}$ *that respect* $\phi$ *(i.e.,* $\phi$ *evaluates to true when each variable* $f$ *of* $\phi$ *is substituted by true if* $f \in \mathcal{Ist}$ *and by false otherwise.)*

On the basis of above definition the propositional formula $\phi$ of feature model $\mathcal{FM}$ shown in Figure 1 can be stated as

$$(MP \leftrightarrow Ca) \wedge (GPS \rightarrow MP) \wedge \sim(GPS \wedge Ba) \wedge \big((Ba \leftrightarrow Sc) \wedge \sim(Col \vee Hr)\big) \wedge \big((Col \leftrightarrow Sc) \wedge \sim(Ba \vee Hr)\big) \wedge \big((Hr \leftrightarrow Sc) \wedge \sim(Ba \vee Col)\big) \wedge (Me \rightarrow MP) \wedge \big(Me \leftrightarrow (Cam \vee MP3)\big) \wedge (Cam \rightarrow Hr)$$

Valuations for which this formula is true characterise the valid instances. Here, a possible instance $\mathcal{Ist}$ is the valuation that assigns true to $\{MP, Ca, Sc, Hr, Me, Cam\}$ and false to $\{GPS, Ba, Col, MP3\}$.

## 3 Related Work

### 3.1 A quality model for evaluating feature models (FMQ)

The FMQ is based on the quality factors, quality attributes, indicators and metrics related to feature models and their development (Thörn, 2010).

FMQ has iterative nature and consists of two iterations. The first iteration is based on the feasibility study of existing quality models while this quality model is used in second iteration for validation and modification (Thörn, 2010).

*Qualities factors and attributes:* The initial proposed model for feature model quality contains six top-level quality factors, with 25 attributes effecting quality factors (Thörn, 2010). This quality model is based on the following quality factors and their attributes.

**Table 1** Quality factors and attributes.

| Factors | Attributes |
|---|---|
| **Changeability** | Adaptability, Extensibility, Stability |
| **Reusability** | Modularity, Self-containedness |
| **Formalness** | Analyzability, Conformance, Consistency, Testability |
| **Veracity**<br>**Accuracy** | Redundancy, Completeness, Reliability, Robustness |
| **Mobility** | Installability, Interoperability, Portability |
| **Usability** | Complexity, Understandability, Learnability, Structuredness, Acceptability, Accessibility, Communicativeness, Visibility |

### 3.2 A business maturity model of SPLE

The purpose of business maturity model of software product lines is to create a strategy for the assessment of the business elements of software product line process (Ahmed and Fernando, 2011).

*Reactive (Level-1):* The "reactive" stage of the business represents the organization not having a stable and organized environment for software product line.

*Awareness (Level-2):* In the beginning of this level, the organization is not keeping with the business practices of product line engineering, however willing to follow rules in coming stages.

*Extrapolate (Level-3):* The organization is ready to gather and circularize market info. The organization makes product line as a region of formal business coming up with.

*Proactive (Level-4):* An organization at this level has been able to establish coordination between business ways and therefore the software product line.

*Strategic (Level-5):* The market size of the product line has increased over a period of time and organization has established and maintained a position as a solution provider in the consumer market.

The difference between the discussed techniques with ours is that we focus on the presence of errors in feature models. Our framework will help designers of SPL to judge the level of individual feature models rather than the whole SPLE process.

## 4 Feature Model Errors

We categorize feature model errors into three groups: inconsistencies, anomalies, and redundancies.

### 4.1 Inconsistencies

Inconsistency arises due to the conflicting information in a feature model. It is impossible to obtain any valid instance from inconsistent feature models. So, inconsistencies are characterized as critical error (Maßen and Horst, 2004). Following are the inconsistency based errors.

*1. Void feature models:* A void feature model defines no instance, i.e., no feature can be selected. This means that each feature is dead including the root. Thus we say that a void feature model is the one whose root is a dead feature (Trinidad et al., 2008). In Fig. 2, some of the void feature models are presented.


**Fig. 2** Examples of void feature model (Danilo, 2008).

*2. Invalid Product:* Invalid product means that invalid instance of a feature model (Benavides, 2010). Invalid instance misses at least one required feature, e.g., mandatory feature of a feature model (Segura et al., 2010). In Fig. 3, a mandatory feature *E* cannot be chosen due to the presence of implies constraint on multiple features that depicted under one alternative set.

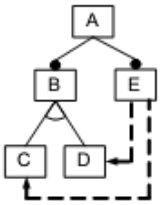


**Fig. 3** Feature model with invalid products (Trinidad et al., 2008).

### 4.2 Anomalies

Anomalies cause improper configuration of instances from a feature model. Anomaly based errors arise due to unrealistic modeled information and this unrealistic information is caused by wrongly captured domain. Variable features are normally selectable, but due to anomalies it might be difficult to select variable features (Maßen and Horst, 2004). Following errors can be categorized as anomalies.

*1. Dead Feature:* A dead feature is a feature that does not appear in any of the instance of the feature model (Benavides, 2007; Trinidad et al, 2008). In Fig. 4, the most commonly occurring cases of dead features are being depicted.

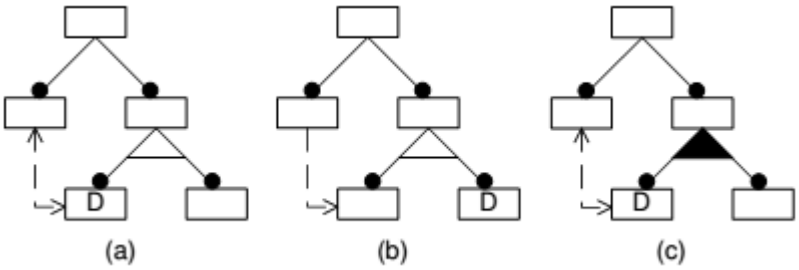


**Fig. 4** Examples of dead features (Segura et al., 2010).

*2. False Variable Feature:* A variable feature is false variable feature, if it has to be chosen whenever its parent is selected in an instance (Trinidad et al., 2008; Zhang and Lin 2011). False variable feature normally found together with dead features (Trinidad et al., 2008). False variable features are also referred as full-mandatory feature (Trinidad et al., 2008). In Fig. 5, most commonly occurring cases of false variable features are shown.

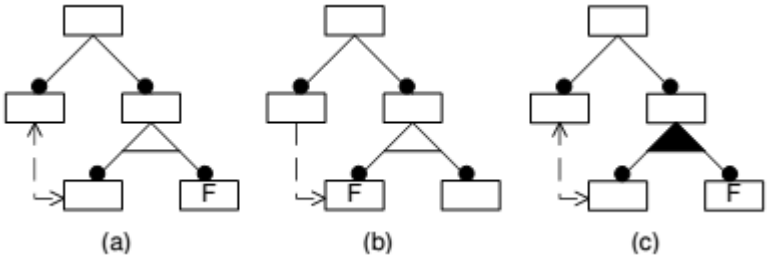


**Fig. 5** Examples of false variable features (Trinidad et al, 2008).

*3. Conditionally Dead Features:* If a feature becomes dead due to the selection or rejection of another feature, it is said to be conditionally dead feature (Hemakumar, 2008) In Fig. 6, an example of conditionally dead feature is depicted.

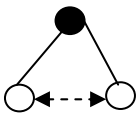

**Fig. 6** Examples of conditionally dead features.

### 4.3 Redundancy

A feature model contains redundancy based errors, if the information in a model is depicted in multiple ways. Developers can interpret a redundantly modelled feature model in multiple ways so, redundancy is considered to be less severe issue (Maßen and Horst, 2004). Normally, redundancies are considered collectively, but here we have defined most commonly occurring redundancies separately which can affect the quality of a feature model. Following errors are considered as redundancy based errors.

*1. Multiple Exclusions*

In multiple exclusions a feature will be excluded by multiple features. In Fig. 7, an optional feature is excluded by multiple mandatory features.

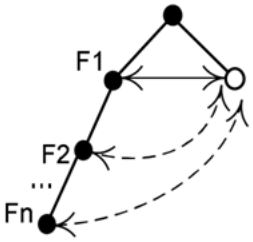


**Fig. 7** Examples of multiple exclusions on a feature (Maßen and Horst, 2004).

*2. Duplicate Feature*

A duplicate feature is a feature appearing multiple times in a feature model.

*3. Multiple Implications*

In multiple implications, a feature will be implied by multiple features. For example, an optional feature is implied by multiple mandatory features, as shown in Fig. 8.

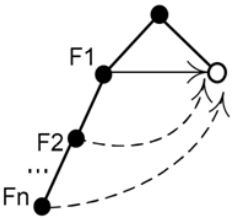


**Fig. 8** Examples of multiple implications on a feature (Maßen and Horst, 2004).

*4. Implied Mandatory Feature*

A mandatory feature appears in all instances of a feature model. If a mandatory feature implied by another feature, it will result in redundancy. Fig. 9 contains some examples of implied mandatory features.


**Fig. 9** Examples of implied mandatory feature (Maßen and Horst, 2004).

## 5 FODA Maturity Model

In this section, we discuss the details about the levels of maturity model for feature oriented domain analysis.

### 5.1 Levels of Maturity Model

In this section, we discuss the details about the levels of maturity model for feature oriented domain analysis. Each level contains some errors of feature model. The arrangement of errors on different levels depends on their severity. A feature model will be of certain quality, if it contains the errors defined on that level and free from the errors listed on the previous levels. Each level has prerequisite which should be met in-order to qualify for that level.

Maturity model for feature oriented domain analysis is shown in Figure 10. We discuss the levels mentioned in Fig. 10 in the following way:

*Step 1:* Definition of level which contains discussion about the errors it contains

*Step 2:* Pre-requisite of this level. We discuss the errors which should not exist in a feature model to be able to be at a certain level

*Step 3:* Explanation by using example. In this step, we use examples to check the above discussed steps for each level. It is worth mentioning that these examples are taken from the published articles.

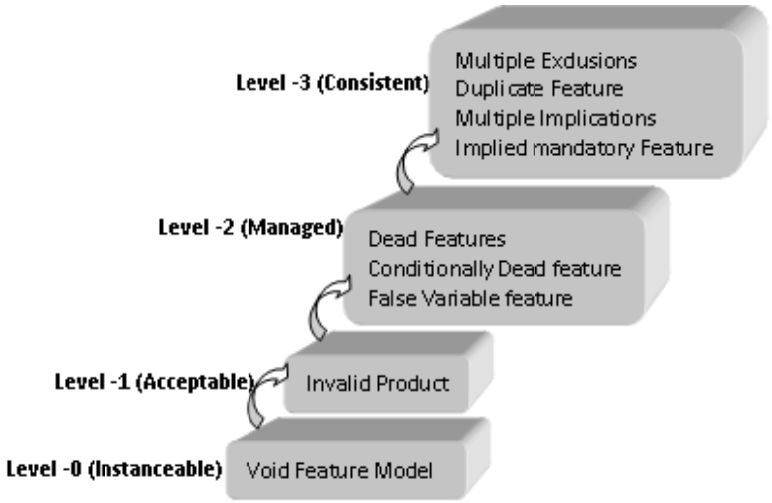


**Fig. 10** Maturity model for FODA.

Let us now discuss the levels.

*1. Instanceable (Level – 0)*

The basic property of a feature model is to produce at least one instance. If a feature model cannot generate any instance is said to be a void feature model (Batory et al., 2006; Trinidad et al., 2008). It is inconsistency based error and most severe among all errors. The quality of feature model will be of level-0 if it is void as mentioned in the maturity model shown in Fig. 10. No product can be developed by Level-0 feature models as a result these models should not be considered for any system.

Normally, first step in feature modelling is to find features and represent them in a tree like structure and secondly, depict cross tree constraints if required. These kinds of errors occur due to the wrong cross tree constraints and contradictory information (Maßen and Horst, 2004).

*Prerequisite:* As this is the basic level so there are no preconditions for this.

*Example:*

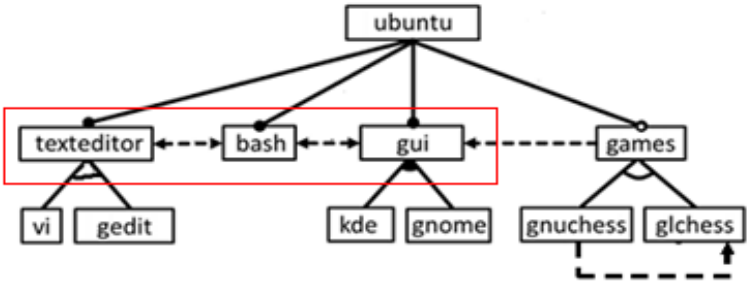


**Fig. 11** Example of void feature model (Felfernig et al., 2013).

In Fig. 11, features *texteditor, bash* and *gui* could not be chosen due to the presence of exclusion constraints between them, whereas the feature *games* will never get selected because of implies constraint with *gui*. On the other hand *vi, gedit, kde, gnome, gnuchess* and *glchess* are not selected because the selection of their respective parent features is not made. Only one feature, i.e., *ubuntu* can only be selected in the instance, this shows that this model is void feature model so; it lies at level-0.

*2. Acceptable (Level-1)*

After finding that a given feature model is void feature model or not, the next step is to check that whether the instances generated by a feature model leads to valid products or not. A valid product of a feature model should contain all mandatory features (Benavides, 2010). The quality of feature model is said to be of level-1 of it constrains invalid product error as per maturity model in Fig. 10.

This is also inconsistency based error which is caused by the use of wrong cross tree constraints and contradictory information. Due to inconsistency, invalid product error also has higher severity level so, placed at this level.

*Prerequisite:* In-order to qualify for acceptable level a feature model should not be a void feature model. A given feature model should produce instances regardless of any errors.

*Example:*

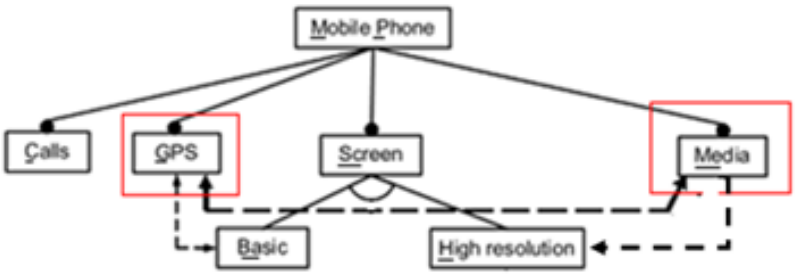


**Fig. 12** Feature model with invalid product (Benavides, 2010).

Feature model shown in Fig. 12 is not a void feature model, but contains the error of invalid product because two mandatory features *GPS* and *Media* can not be selected due to exclusion constraint between them. To get a valid product all mandatory features should be chosen in an instance. As the feature model shown in Fig. 11 produces invalid product; hence it has level-1 quality.

*3. Managed (Level-2)*

The severity level of anomaly based error is less than inconsistency and higher than redundancy based errors (Maßen and Horst, 2004). So, anomaly based errors placed on this middle level. The maturity level of a feature model will be managed if it contains any of the anomaly based error mentioned in maturity model of Fig. 10.

Anomaly based errors includes dead feature, conditionaly dead feature and false variable feature. Most of the time false variable feature (full-mandatory feature) appears with dead feature (Trinidad et al., 2008) that is why false variable feature error is also considered at this level along with dead features.

*Prerequisite:* To qualify for quality level-2 (Managed) a feature model should not be void and also generate valid instance (all mandatory features should be instantiated).

*Example:*

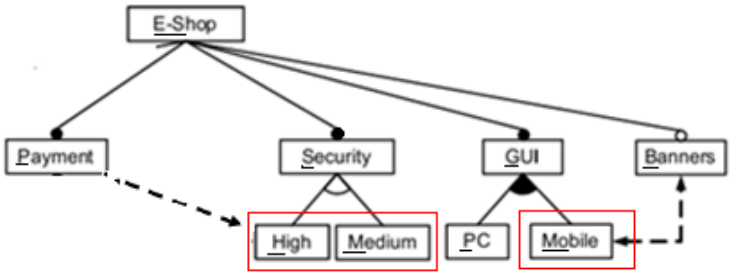


**Fig. 13** Feature model with dead, false variable features and Conditionally dead features (Segura et al., 2010).

The quality of feature model depicted in Fig. 13 is of level-2 because it contains a dead feature *Medium,* a false variable feature *High* and a conditionally dead feature *Mobile*. The feature *High* is false variable feature because of implies constraint by a mandatory feature *Payment* and simultaneously feature *Medium* will not be selected due to this implies constraint and also both *High* and *Medium* lies under same alternative set.

Note that the feature *High* is variable, but behaves like mandatory (a false variable feature) due to the presence of implies constraint by *Payment* (a mandatory feature).

In Fig. 13, if the feature *Banners* is selected then *Mobile* will become a dead feature, this error is based on the selection or rejection of an optional feature *Banners* so, it is called conditionally dead feature as a result this feature model on maturity level-2 (quality wise).

*4. Consistency (Level - 3)*

After inconsistency and anomaly the next severity level is of redundancy. Redundancy based errors have low severity level (Thörn, 2010). Redundancy based error leads to misinterpretation for the developers and as a result low quality product. Most commonly occurring redundancy based errors are multiple exclusions, duplicate feature, multiple implications and implied mandatory feature. A feature model will be on this maturity level (consistent), if it contains redundancy based errors as per maturity level depicted in Fig. 10.

*Prerequisite:* To qualify for this level, a feature model should be free from all errors mentioned in the previous levels i.e. instanceable, acceptable and managed.

*Examples:*

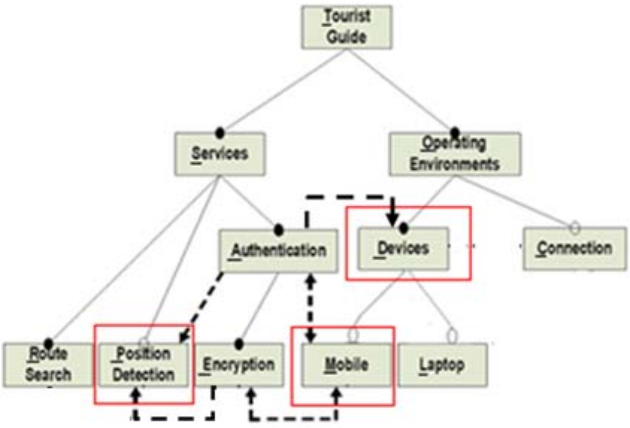


**Fig. 14** Feature model with multiple exclusions, multiple implications & implied mandatory (Zhang and Lin, 2011).

In Fig. 14, the feature *Position Detection* implied by two mandatory features *Route Search* and *Authentication* that results in multiple implication error (Maßen and Horst, 2004). While a feature *Mobile* is excluded by two mandatory features *Authentication* and *Encryption,* which causes multiple exclusions error (Maßen and Horst, 2004). Similarly, a mandatory feature *Devices* is implied by another feature *Authentication* which results in implied mandatory feature error (Maßen and Horst, 2004). Feature model depicted in Fig. 14 contains redundancy based errors hence; its maturity level is consistent.

Feature model of Fig. 15 contains two features with the same name *Java Support* results in duplicate feature error that is causing confusions for developers while referring to these duplicate features. Due to redundancy this feature model is at consistent level.

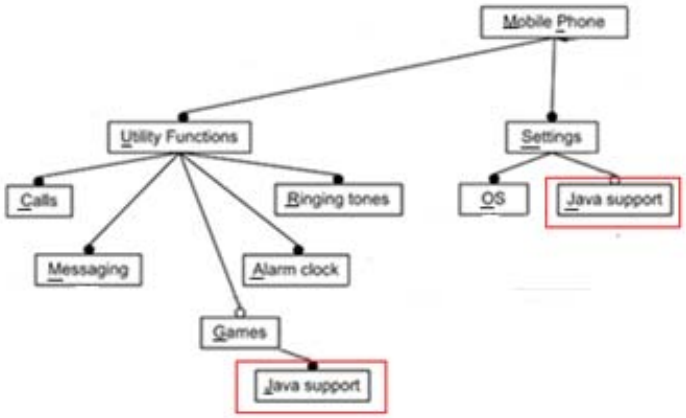


**Fig. 15** Feature model with duplicate features (Segura, 2011).

## 6 Conclusion and Future Work

In this paper, we have shown a framework to measure the quality of feature models. In our framework, we have provided five different levels which are based on the errors in feature models. Our framework will help developers to know the quality level of feature models. We have described all the levels with the help of different examples.

Future work will mainly focus on the development of the algorithms for each level. Our plan is to automatically check the quality level of a given feature model.

## References

Ahmed F, Fernando C. 2011. A business maturity model of software product line engineering. Information Systems Frontiers, 13(4): 543-560

Batory D, Benavides D, Antonio R. 2006. Automated analysis of feature models: challenges ahead. Communications of the ACM, 49(12): 45-47

Batory D. 2005. Feature models, grammars, and propositional formulas. Springer, Berlin, Heidelberg, 1: 7-20

Benavides D, Segura S, Ruiz-Cortés A. 2010. Automated analysis of feature models 20 years later: A literature review. Information Systems, 35(6): 615-636

Benavides D. 2007. On the Automated Analysis of Software Product Lines Using Feature Models. Dissertation, Universidad de Sevilla, Spain

Böckle G, Van Der Linden F. 2005. Software Product Line Engineering (Pohl K, ed) (Vol. 10). Heidelberg, Springer

Clements P. 2001. Software Product Lines: Practices and Patterns. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA

Felfernig A, David B, Galindo J, Reinfrank F. 2013. Towards Anomaly Explanations in Feature Models. In: Proceedings of the Proceedings of the 15th International Configuration Workshop (ConfWS-2013). 117-124

Hemakumar, A. (2008, September). Finding Contradictions in Feature Models. In: Proceeding of Workshop on Analyses of Software Product Lines (ASPL 2008), collocated with 2008 International Software product Line Conference (SPLC 2008). 183-190

Kang K, et al. 1990. Feature-oriented Domain Analysis (FODA) Feasibility Study. Technical Report,

Carnegie-Mellon University Pittsburg, SEI, USA

Maßen T and Horst L. 2004. Deficiencies in feature models. In: Proceedings of the Workshop on Software Variability Management for Product Derivation - Towards Tool Support, collocated with the 3rd International Software Product Line Conference (SPLC'04) (Vol 3154)). 331, Springer, Berlin, Heidelberg

Mendonça M. 2009. Efficient reasoning techniques for large scale feature models. Diss. University of Waterloo, Canada

Naeem M. 2012. Matching of Service Feature Diagrams based on Linear Logic. Dissertation, Department of Computer Science, University of Leicester, UK

Rosso, CD. 2006. Experiences of performance tuning software product family architectures using a scenario-driven approach. In Proceedings of the 10th International Conference on Evaluation and Assessment in Software Engineering (EASE 2006): 30~39

Rubin J, Chechik M. 2012. Combining related products into product lines. In Proceedings of the 15th international conference on Fundamental Approaches to Software Engineering (FASE'12) (Lara J, Zisman A, eds). 285-300, Springer-Verlag, Berlin, Heidelberg

Segura S, Benavides D, Ruiz-Cortés A. 2010. FaMa Test Suite v1.2. Technical Report ISA-10-TR-0. 1-52, Applied Software Engineering Research Group, University of Seville, Spain

Segura S. 2011. Extended Support for the Automated Treatment of Feature Models. Dissertation, University of Sevilla, Spain

Thomas E. 2008. SOA: Principles of Service Design (Vol 1). Prentice Hall, Upper Saddle River, USA

Thörn, C. 2007. A Quality Model for Evaluating Feature Models. In Proceedings of second volume (Workshops) of the 11th International Software Product Lines Conference. 184-190, Kindai Kagaku Sha Co. Ltd., Tokyo, Japan

Thörn C. 2010. On the Quality of Feature Models. Dissertation, Department of Computer and Information Science, Linköping University, Sweden

Trinidad P, et al. 2008. Automated error analysis for the agilization of feature modeling. Journal of Systems and Software, 81(6): 883-896

Zhang G, Ye H, Lin Y. 2011. Feature model validation: A constraint propagation-based approach. In: 10th International Conference on Software Engineering Reaserch and Practice. Las Vegas, USA