Sindh Univ. Res. Jour. (Sci. Ser.) Vol 48(4) 873-878 (2016)

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

On the Use of Meta Model for Error Free Feature Modelling

Z. RASHID, M. NAEEM*, A. IQBAL, N. H. ARIJO++**, H. A. WAHAB***, M. JAVED, F. HUSSAIN*, F. BAHADUR

Department of Information Technology, Hazara University, Mansehra

Received 23rd December 2015 and Revised 19th August 2016

Abstract: Feature model is an efficient modelling technique to generate families of software products. The presence of errors in feature models degrades the quality of software products. There are two established ways to handle errors, i.e., removal and avoidance. Error avoidance is more economical as compared to error removal in software engineering. In this paper we are propos ingmetamodel based error avoidance technique for feature models.

Keywords: Quality of Feature Model, Meta Model**,** Software Product Line, Instance, Feature Diagram

1. INTRODUCTION

The huge production needs standardized processes, especially in the case of products sharing features (Czarnecki, 2000). Since the last decade, software systems are getting complex. Their expected time-to-market for introducing new changes become challenging, whereas expected usage period has grown. In order to address these issues in software engineering, software engineering has provided different approaches for reusability, adaptability, flexibility, and control of performance and complexity of software. Several software paradigms about the expected effects of reusability in terms of reducing development effort and time as well as improving quality of the software characteristics has started. Object oriented modeling; multi-tier architectures, process maturity, component technology and other approaches lead to progress in evolve ability and efficiency (Mendonca, *et al* 2008).
Producing things at large scale requires standardized processes, especially for the similar products or product line. Productions are being organized by companies in large amount of production (Mendonca, *et al* 2008). The software systems having similar specifications guide us towards Software Product Line (SPL)(Kang, *et al* 1990). By considering commonality and variability in the software families in a systematic way leads us to SPL Engineering (Kang, *et al* 1990, Javed, *et al* 2014, Clements and Northrop, 2001, Naeem, 2012).

Software products, in SPL are being created by reusability (Czarnecki, *et al* 2004a, Czarnecki, *et al* 2004b). Features represent aspects of software (Czarnecki, *et al* 2002). Feature model depict the relationships and constraints on of these features (Korson and McGregor, 1990). Feature models are tree like structures having root and child nodes that describe successive refinements of the variability in a software product-line.

Feature modeling is getting importance among researchers' academia and industry. There are a lot of problems in the feature models which need to be resolved to produce a good quality model (Batory, *et al* 2006). One of those problems is the presence of defects in feature models. There are different available techniques for the defect identification and correction in feature modeling (Mendonca, *et al* 2008, Rincón, *et al* 2014). Error removal and error avoidance are two techniques of addressing the errors. Error removal is expensive technique than error avoidance. Further, an error avoidance technique is not only cost effective, but provides high quality feature models as well.

(Fig.1) shows the origin of the work. Software Reuse is the main goal of software development and Software Product Lines is a modern technique for the software reuse. Feature models are used to model the comonality and variability in SPLs. Our current effort is related to error avoidance that is a contribution to develop a reliable and efficient feature model.

Fig. 1 Origin of work

We have proposed a rule based approach for error free feature models written in natural language in our previous paper (Zahid, *et al* 2015). Although the approach is covering all the errors in feature models, but we want those rules to be realized by a formal framework.

++Corresponding author Niaz Hussain Arijo email: niaz_arijo@hotmail.com
*Abbottabad University of Science & Technology, Havelian, Abbottabad
**Institute of Information & Communication Technology, University of Sindh, Jamshoro
***Department of Mathematics, Hazara University, Mansehra

In this paper, we focus on development of a framework for error free feature models based on Meta model. All feature models should be typed with this Meta model and these models will not be affected by the errors. The remaining part of the paper is arranged as: Section 2elaborates on the basics of feature modelling. In Section 3, we discuss related approaches, while the Section 4 covers the proposed methodology. Section 5 gives concludes and provides quick look to our future plans.

2. BACKGROUND

A. FEATURE MODEL

Feature models were coined by a research group at Software Engineering Institute in the form of a report called Feature Oriented Domain Analysis (FODA)(Kang*et al* 1990). A feature model represents a tree like structure capturing common and variable features in software product lines (Kang, *et al* 1990), while we use the notations proposed in (Czarncki, 2000) for feature modeling. For example, **(Fig. 2)** depicts a feature diagram FD for an online service of a travel guide.

An instance of FD is the permissible selection from the set of features of FD. For example, a permissible selection from the travel service of Fig. 2 must realize the follow: 1) the root of the feature diagram must be the part of each instance; 2) a feature can only be selected, if and only if its parent feature is already selected; 3) a mandatory sub-feature of a selected feature must be the part of that instance; 4) one feature from the Alternative-group of a selected feature must be selected; 5) the feature selection must be made from the Or-group of already selected feature.

Apart from the above conditions, features can be connected by horizontal constraints: 1) Requires constraint: the target feature of requires constraint (shown by dashed arrow in Fig. 2) must be chosen if its source is chosen; 2) Excludes constraint: Both the source and the target of exclude constraint (not depicted in Fig. 2) must not be the part of an instance.

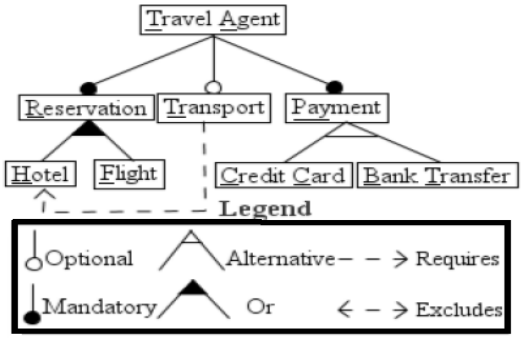

Fig. 2 Feature model of an on-line travel guide

Therefore, {TA, Res, H, Pay, BT} represents an instance of FD}. The notion of instance can also be realized at the level of Classical Logic (Czarncki and Wasowski, 2007). For example, a classical formula of the feature diagram in **(Fig. 2)** is:

(TA↔Res) ^ (Res↔(HVF)) ^ ((Tr→Ta) ^ (Tr→H) ^ (TA↔Pay) ^ (CC↔(~BT^Pay)) ^ (BT↔(~CC^Pay))

All those valuations which leads to true values provides valid instances In our case, a valid can be obtained if we assign true to {TA, H, Pay, BT, Res} and false to {Tr, CC, F}.

The presence of errors effects the quality of feature models. Let us know explain the available errors in the following section.

B. ERRORS IN FEATURE MODEL

According to the available literature, there are three types of errors included in the feature models (Javed, *et al* 2014).

1) Inconsistency

Conflicting information in feature models leads to inconsistencies. Inconsistency based errors are marked as crucial because consistent instances can not be obtained from such models (Maßen and Lichter, 2004). Inconsistencies include the following errors:

Void Feature Models:

Void feature model cannot produce any instance. These feature models are suffered from existence of crosstree constraints among mandatory features. Inconsistent, insatiable or invalid feature models are also used for Void feature models (Hemakumar, 2008). **(Fig. 3)** depicts some possible examples of void feature model.

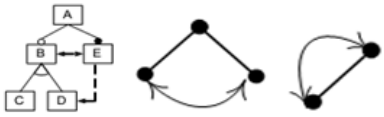

Fig. 3 Void feature models

Invalid Instance Error:

Invalid instance gives invalid output because it does not select the mandatory feature (Trindad*et al* 2008). The **(Fig. 4)** below contains some example of invalid instance error.

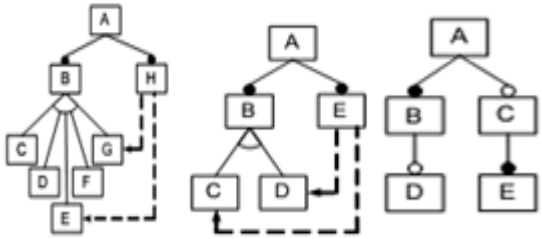

Fig: 4 invalid Instance Error

2) Anomalies

These errors arise due to modelling of unrealistic information. None of the proper instances can be obtained from a feature model containing anomalies. The severities of such errors are considered as medium level (Maßen and Lichter, 2004).

Dead Feature:

This is a kind of a feature that could not be selected in any of its the instances (Trindad, *et al* 2008). Dead features are the result of wrongly placed horizontal constraints and it makes the feature model void (Segura, 2008). **(Fig. 5)** contains the example of a dead feature.

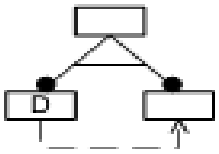

Fig. 5 Dead Feature

False variable feature:

A feature that is variable but appears to be mandatory is called false variable feature, i.e. a feature that is modelled as variable but it always get selected along with its parent(Trinidad, *et al* 2008). Usually, these features appear along with dead features in a feature model (Trinidad, *et al* 2008). **(Fig. 6)** depicts the example of false variable feature.

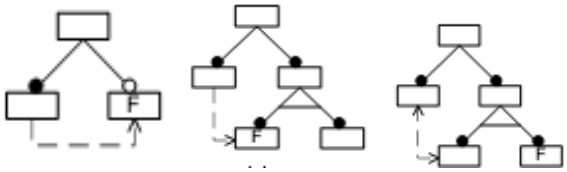

Fig. 6 False Variable Feature

Conditionally Dead Feature:

Conditionally dead feature is like a dead feature but it become dead on an optional condition. It becomes dead because of either selecting or rejecting another feature (Hemakumar, 2008). **(Fig. 7)** below depicts conditionally dead feature.

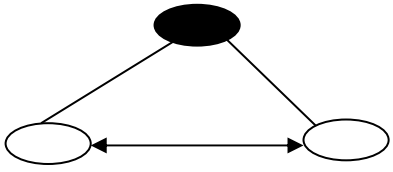

Fig. 7 Conditionally Dead feature

3) Redundancies.

Redundantly modelled feature models means that information is modelled in multiple ways. These errors are considered to be a least severe among all errors (Maßen and Lichter, 2004). Redundancies includes:

More than One Exclusion:

This error happens when more than one mandatory features excludes a feature (Maßen and Lichter, 2004). For example, the **(Fig. 8)** below suffers from this error because a variable feature (optional feature) is excluded by more than one core features.

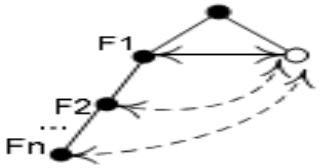

Fig. 8 Multiple Exclusion

Duplicate Features:

This means that a feature model contains more than one feature with the same name (Javed, *et al* 2014). In **(Fig. 9)**, feature E is a duplicate feature.

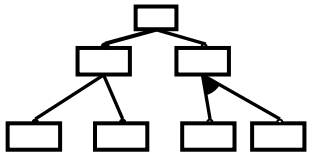

Fig. 9 Duplicate FeatureMultiple Implications:

This error happens when more than one mandatory features requires a feature (Maßen and Lichter, 2004). For example, the **(Fig. 10)** below suffers from this error because a variable feature is required by more than core features.

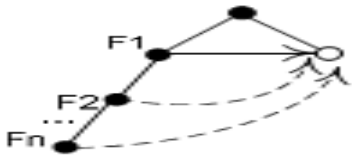

Fig. 10 Multiple implications

Implied Mandatory Feature:

This error exists if a mandatory feature is connected with another feature by requires constraint(Javed, *et al* 2014). **(Fig. 11)** contains some examples of implied mandatory feature.

Fig. 11 Imply mandatory Feature

3. RELATED WORK.

There are several efforts available in the literature of SPL that covers error handling. Benavidas *et al.* proposed the usage of constraint programming for analyses on feature models (Benavidas *et al* 2004, Benavida s*et al* 2005a, Benavidas *et al* 2005b). They

interpreted feature models into CSP by the help of mapping rules. The authors' approach provides analysis as well as the optimization of extended feature models. Benavides et al. also developed a tool for such analysis (Benavidas *et al* 2007, Trindad *et al* 2008). Furthermore, they have also provided comparative analysis of available solvers for the analyses of feature models(Benavides, *et al* 2006a, Benavides, *et al* 2006b, Segura, 2008).

Trindad *et al.,* in (Trindad *et al* 2006, Trindad *et al* 2008) worked based on diagnosis theory of Reiter and focus on the identification and discussion of defects in feature models (Reiter, 1987) and constraint programming.

Rincon proposed a technique for the analysis of dead and false variable feature by the help of graphical rules (Rincón*et al* 2014). Author proposed rule-based approach to identify dead and false optional features, identify certain reasons of such errors and discussed them in natural language helping modelers to correct found defects.

They have borrowed 31 feature models from the published literature for the analysis and evaluation of their approach. Further, they concluded that proposal is effective, accurate and scalable to 150 features (Rincón*et al* 2014).

All these techniques are analysis of the errors after their existence which can only be detected and then removed. But, there arise a question, why we don't avoid the errors in feature model at the level of design? There is no such technique that stops errors from getting its way into the feature models.

This needs a complete analysis of the errors which are possible to avoid and making a complete framework of the rules according to which the errors can be minimized. We have worked on the error avoidance in our past paper in which we propose 10 rules for the error avoidance of the feature model (Zahid, *et al* 2015).

4. METHODOLOGY

In order to get error free feature models, we proposed therules in (Zahid, *et al* 2015).For the understanding and modeling in the metamodel we further rearrange these rules.

Furthermore rule 5 is a general concept for the mandatory feature, when we include local and global mandatory feature then this rule has no need in this work because designers have to see level of the mandatory feature in design time to decide mandatory for its parents.

In the same way rule 1 also overlap rule 5 where designer have to decide the level of the feature to model it in mandatory or in optional, if mandatory the level of mandatory from rule 5.

A. Meta model

For the modeling these rules in Meta model we use following methodology.

1. A mandatory feature must not be the part of any exclude constraint.

For the modeling of this rule in Meta model we kept restriction on the constraint that if core condition is true or opt is false then both (source and target) constraint will be disallowed.
This will disallow any constraint for the mandatory feature as well as core feature.

2. A mandatory feature must not be the part of any implies constraint.

Same as in rule 1 we kept restriction on the constraint that if core condition is true or opt is false then both (source and target) constraint will be disallowed.

3. In alternative (exclusive OR) relevance, the sibling features should not have implies constraint.

For the modelling of this rule in metamodel we kept the restriction on gType. If gType is Alternative that constraint will be disallowed.

4. If a parent feature excluded then no constraint is possible between them and their children across the subtrees.

In the metamodel we address by using array with all the features having the detail of exclusion. When parent have exclude constraint between then their children well keep the record of the parent exclusion such that the exby array of the parent, that have the name of the feature by which that is excluded will be copied in their children exby.

5. The names of the features must be unique.

In the metamodel we kept a restriction while making new feature all feature must be checked. If same name exist in the feature then feature will be disallowed.

6. In implies cross tree constraint, child of source feature should not imply target feature.

For this rule we keep an array named srcOf that is the names of target features. For every child srcOf of its parent will be copied in its srcOf. This every child have detail about the implies child have no need to imply the target of parent.

7. In implication a target feature should have one path from source feature.

In the transitive implication when a feature is implied it become the part of Srcof of source feature and soruce feature become trgOf of that feature. But when source and/or target already have any other constraint then this function of metamodel is needed.

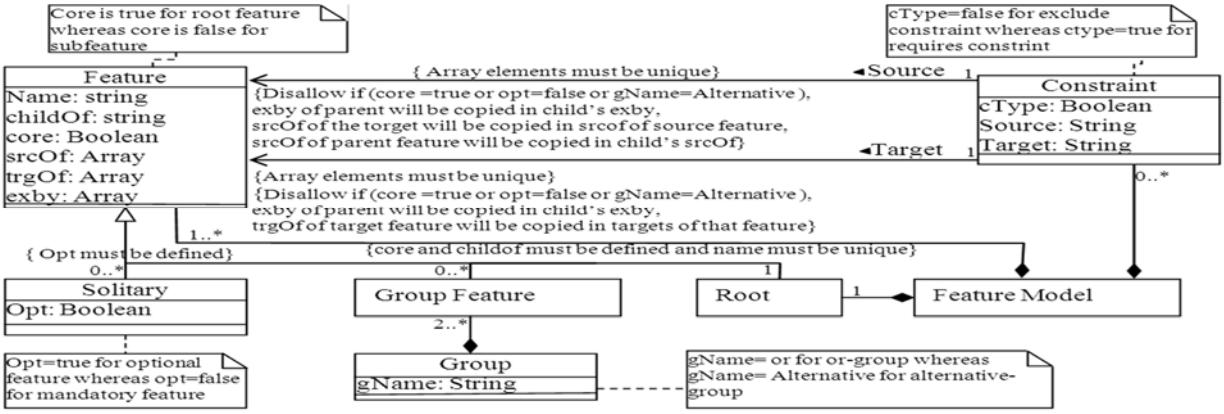

Fig. 21. Meta model of feature models

In this constraint we have to look the srcOf and trgOf of the feature. For imply constraint srcOf of the target feature will be copied in srcOf of source feature and trgOf of the source feature will be copied trgOf of target feature. As well as in all target feature in their line.

For all constraint there is a by default check that if the name of the feature is already present then don't overwrite that array. All the element of arrays will be unique and they will exit only one time in the array.
In order to present the rules in formal method we have presented Met model. This model is based on the seven rearranged rules presented above.

Rule1 and 2 are covered in the Metamodel by the constraints that source and target are not allowed in the mandatory feature. In the same way rule 3 is covered in the constraint that if the group type is alternative then don't allow source and target.
Error due to developer mistake: These errors are also possible due to developer mistake which cannot be handle using this framework. List of mistakes by the developer is as under:

1. Keeping mandatory in the list of optional feature.
2. Keeping mandatory feature in list of optional feature.
3. Keeping global feature's parent mandatory.
4. Designing condition wrongly.

Error coming caused by these mistakes cannot be handled through this framework. This need complete analysis of the feature type.

4.2 Feature Modeling Procedure

Although whole concept is defined in the metamodel but cross tree constraints are very important so we describe the modeling procedure of the cross tree constraints also in natural language. Following is the description of the source and target conditions used in metamodel.

4.2.1 Source Conations:

Condition I. Disallow if (core =true or opt=false or gName=Alternative)

- This condition in metamodel avoid exclusion of core feature mandatory feature and alternative feature
- Covers Void feature model, invalid instance, dead feature and false variable feature.

Condition II. Exby parent will be copied child's exby

- Record of the exclusion of the parent will be copied in the record of the children such that all the children have the detail of exclusion of the parent.
- This covers the error of multiple exclusion, dead feature(implied).

Condition III. srcOf of the torget will be copied in srcof of source feature.

- Record of the imply and implied will be copied in the relevant feature i.e sorce of source feature.
- This covers the error of multiple imply.

Condition IV. srcOf of parent feature will be copied in child's srcOf.

- The record of imply of the parent feature will be copied in the record of the child.
- This also covers the multiple implication.

1) Target Conditions:

Condition V. Disallow if (core =true or opt=false or gName=Alternative)

- This condition prevents the conflict the source and target.

Condition VI. Exby of parent will be copied in child's exby

- Record of the exclusion of the parent will be copied in the record of the children such that all the children have the detail of exclusion of the parent.
- This covers the error of multiple exclusion, dead feature (implied).
- Keep in mind that there will be two feature in one constraint having the array of exby array, same process will be repeated in both source and target side.

Condition VII. Trgof of target feature will be copied in targets of that feature.

- In this step record of the implied feature will be copied in the record of the target feature of that feature.
- This covers the error of multiple imply through transitive implication.

5. CONCLUSION

In this paper, we developed a meta model for error free feature models. Our approach avoids entry of errors in feature models. In future we shall work to develop a tool support for designing error based on error avoidance in the feature models, to produce error free feature model.

REFERENCES:

Batory. D, D. Benavides, and A. Ruiz-Cortes. (2006) Automated analysis of feature models: challenges ahead. Communications of the ACM 49.12: 45-47.

Benavides. D, A. Ruiz-Cort´es, and P. Trinidad. (2004) Coping with automatic reasoning on software product lines. In Proceedings of the 2nd Groningen Workshop on Software Variability Management.

Benavides. D, A. Ruiz-Cort´es, and P. Trinidad. (2005a) Automated reasoning on feature models. In 17thCAiSE Conference, Vol 3520 of LNCS, 491–503.

Benavides. D, A. Ruiz-Cort´es, and P. Trinidad. (2005b) Using constraint programming to reason on feature models. In The 17thSEKE 2005, 677–682.

Benavides. D, S. Segura, P. Trinidad, and A. Ruiz-Cort´es. (2006a) A first step towards a framework for the automated analysis of feature models. In Managing Variability for Software Product Lines.

Benavides. D, S. Segura, P. Trinidad, and A. Ruiz-Cort´es. (2006b) Using java CSP solvers in the automated analyses of feature models. LNCS, 4143:389–398.

Benavides. D, S. Segura, P. Trinidad, and A. Ruiz-Cort´es. (2007) FAMA: Tooling a framework for the automated analysis of feature models. In Proceeding of VAMOS, 13`–134.

Czarnecki. K, S. Helsen, and U. Eisenecker. (2004a) Staged configuration using feature models. In Proceedings of SPLC 2004, Vol 3154 of LNCS.

Czarnecki. K, S. Helsen, and U. Eisenecker. (2004b) Formalizing cardinality-based feature models and their staged configuration. Tech. Report, Deptt of Elec. & Computer Engg., University of Waterloo, Canada.

Czarnecki. K, T. Bednasch, P. Unger, and U. Eisenecker.(2002) Generative programming for embedded software: An industrial experience report. In Proceedings of GPCE 2002, Vol 2487 of LNCS

Clements. P and L. Northrop. (2001) Software Product Lines: Practices and Patterns. : Addison-Wesley.

Djebbi. O, C. Salinesi, and D. Diaz. (2007) Deriving product line requirements: the red-pl guidance approach. In 14th APSEC, 494–501.

Hemakumar, A. (2008) Finding Contradictions in Feature Models. In SPLC (2), 183-190.

Kang K, S. Cohen, J. Hess, W. Nowak, and S. Peterson. (1990) Feature-oriented domain analysis (FODA) feasibility study. Tech. Report, SEI, Carnegie Mellon University, Pittsburgh, PA.

Korson. T and J. D. McGregor, (1990) Understanding object-oriented: a unifying paradigm, Communications on ACM, vol. 33, no. 9, 40–60.

Marcilio M., A. Wasowski, K. Czarnecki, D. Cowan. (2008) Efficient compilation techniques for large scale feature models. In Proceedings of the 7th GPCE, 13-22.

Naeem. M (2012) Matching of Service Feature Diagrams based on Linear Logic. PhD Dissertation, Department of CS, University of Leicester, UK.

Reiter R. (1987) A theory of diagnosis from first principles. Artificial Intelligence, 32(1):57–95.

Rincón, L. F., G.L. Giraldo, R. Mazo, and C. Salinesi. (2014) An ontological rule-based approach for analyzing dead and false optional features in feature models. ENTCS, 302: 111-132.

Segura. S, (2008) Automated analysis of feature models using atomic sets. In Proceedings of ASPL, 201–207.

Trinidad. P, D. Benavides, and A. Ruiz-Cort´es. (2006) A first step detecting inconsistencies in feature models. In CAiSE Short Paper Proceedings.

Trinidad. P, D. Benavides, A. Dur´an, A. Ruiz-Cort´es, M. Toro. Automated error analysis for the agilization of feature modeling. Journal of Systems and Software, 81(6):883– 896.

Trinidad. P, D. Benavides, A. Durán, A. Ruiz-Cortés, M. Toro. (2008) Automated error analysis for the agilization of feature modeling. Journal of Systems and Software 81, no. 6: 883-896.

White. J, D. Schmidt, D. Benavides P. Trinidad, and A. Ruiz- Cortes. (2008) Automated diagnosis of product-line configuration errors in feature models. In Proceedings of the Sofware Product Line Conference.

White. J, B. Doughtery, D. Schmidt, and D. Benavides. (2009) Automated reasoning for multi-step software product-line configuration problems. In SPLC, 11–20.

Rashid Z., M. Naeem, F. Bahadur, and M. Javed. (2015) Towards the Development of Error Free Feature Models. SAICON International Conference.