2014

ISSN : 0974 - 7435 Volume 10 Issue 14

BioTechnology

An Indian Journal

FULL PAPER

BTAIJ, 10(14), 2014 [8118-8125]

Variability modeling for software product line

Luo Daizhong[1], Diao Shanhui[2]

[1]School of Software Engineer, Chongqing University of Arts and Sciences, Chongqing, (CHINA)

[2]Department of Education Administrator, Chongqing University of Arts and Sciences, Chongqing, (CHINA)

Luodaizhong@126.com

ABSTRACT

The variability model which captures the commonality and variability of the software product families is very important in domain requirements modeling. In the software product families, it will be difficult to establishing the variability model in the domain analysis. An novel variability modeling method with the extended UML is presented for variability modeling, it not only supports variability type of use cases such as optional, alternative and or, but also supports variability constraint such as variant to variant, variant to variation point and variation to variation point. We present the formal definition of variability type and variability constraint, it helps us to validate the variability model correctness. Finally, we illustrate the variability modeling approach with a mobile phone SPL and then discuss about the formalization of variability type and constraint.

KEYWORDS

Software product line; Variability modeling; Variability type; Constraint, UML.

INTRODUCTION

Software Product Line (SPL) technology has become research spot and has been widely applied in domain engineer. Product line engineering is a development paradigm that explicitly addresses reuses for software development, can effectively promise large gains in productivity, quality and time to market reduction[1].

Software product line is a reuse technology of software architecture. It develops requirement specifications, test cases, components by identifying commonality features of similar productions in the special domain. The development process of software product line includes two key activities, domain engineering and application engineering (Figure 1).

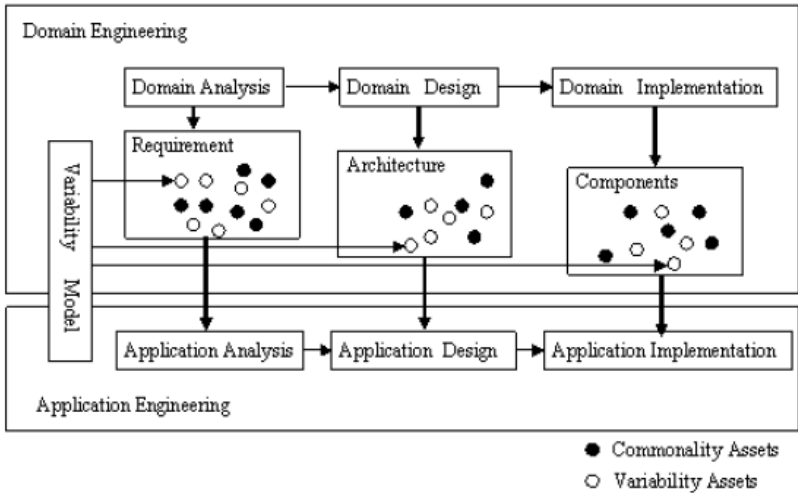

Figure 1 : Development process of Software Product Line

Domain engineering includes three stages of domain analysis, domain design and domain implementation. The primary task of domain engineering is identifying commonality of similar productions in special domain and defining variability of production in software product line, developing domain artifacts that include architecture of software product line, components, documents, requirement specification and test cases etc.

Application engineering constructs a specific product with commonality core assets of software product line which is needed in all application product and variability assets of domain artifacts which is needed in special application product. The same as domain engineering, application engineering includes three stages of application analysis, application design, application implementation.

Variability modeling is a modeling technology for domain engineering, which needs to be integrated into the software engineering traditional modeling technology with UML, but the traditional UML modeling technology do not support for variability modeling. Literatures[3] suggested that the variability models are constructed by model difference and merging. Literatures[4,5,12,13,14] presented an approach of modeling variants according to variant types. Base on the comparison of variability modeling approaches[8], Literature[9] proposed the hierarchical variability modeling for software architectures, Literatures[6,7,10,15,16] discussed the syntax and semantics for variability model, described the formal definition of the model. But all the literatures are incomplete for lack of the dependency, constraint and formalization integrity. To address this problem, we present an approach for variability modeling with the formal definition of variability type and variability constraint.

The remainder of this paper is structured as follows. Section 2 presents variability of software product line which includes variation point, variant, variability type and variability constraint. Section 3 presents the approach of variability modeling with the extended UML language, and formalization of variability type and constraint. Section 4 describes the Mobile Phone case to illustrate the variability modeling approach. Finally Section 5 concludes this paper.

SOFTWARE PRODUCT LINE VARIABILITY

The term variability refers to the ability of change in software product line. Variability means that the decision is delayed to a specific time, and then binds variants according to requirement of application product. The variability in different products is a unique function and specific requirements for application products, so the variability modeling is a key activity of domain engineering. The variability of the software product line involves variation point, variant, variability type and variability constraint.

Variation point

In software product line engineering, a variation point is a representation of a variability subject within domain artifacts enriched by contextual information. Variability subjects and the corresponding variability objects are embedded into the context of a software product line. They represent a subset of all possible variability subjects and a subset of all possible variability objects from the real world, which are necessary to realize a particular software product line[1].

Variant

A variant is a representation of a variability object within domain artifacts[1]. For example, a mobile company wants to build phones in different colours, therefore a variation point "colour of a phone" (phone is the context of the variation point) is defined. If an example mobile company builds white and blue phones, therefore only the variants 'white (phone)' and 'blue (phone)' are defined.

Variability type

The variability type presents the conceptual relationships between the variation point and the variants, which refers to the mandatory or variable form of bind the variants to the variation point. Mandatory defines the variant that must be selected to the variation point, if the variation point is selected. Variable means if a variation point is selected, the variants of the variation point may be selected with the form of *Alternative, Optional, Or, Option Alternative, Optional Or*.

Variability constraints

The variability constraint refers to the valid combinations among variation point, variants. In other words, variability constraints are combination rules when the variants are tailored or bound to the variation point. We identify the two combination rules, i.e., *Requires* and *Excludes*, to capture the constraints between variant to variant, variant to variation point, variation point and variation point. *Requires* define a relationship that the selection of a variant or a variation point *requires* the selection of the other variants or variation points. *Excludes* define a relationship that the selection of a variant or a variation point *excludes* the selection of the other variants or variation points.

Variant to Variant Constraint. For example, variant *requires* variant refers to that a variant is selected, the other variant must be selected. Variant *excludes* variant refers to that a variant is selected, the other variant must be not selected.

Variant to Variation Point Constraint. For example, variant *requires* variation point refers to that a variant is selected, all variants of the other variation point must be selected. Variant *excludes* variation point refers to that a variant is selected, all variants of the other variation point must be not selected.

Variation Point to Variation Point Constraint. For example, variation point *requires* variation point refers to that a variation point is selected, the other variation point must be selected. Variation point *excludes* variation point refers to that a variation point is selected, the other variation point must be not selected.

VARIABILITY MODELING

Software product line modeling must describe the commonality and variability of the software products and their constraint relationships[11]. The commonality of the software products which is stable can directly model with the UML language, but the UML language does not support the variability modeling, so we must extend the UML language to address the problem.

Identifying variability

Variation points and variants are used to define the variability of a software product line. Thus, it is essential to identify variation points and variants. In the following, we provide three basic steps for variability modeling. The first step is to identify varieties in the real world, i.e. identifying the variability subject[1]. The second step is to define a variation point in the context of the software product line. This step is necessary as there is a difference between variability in the real world (represented by variability subjects) and variability in a software product line (represented by variation points). The third step is to define the variants which are necessary to be bound to the variation point, and regards as variants of the variation point.

The requirement analysis of domain engineering is a process for refining requirement and identifying variability. It extracts the domain requirements from the similar software products and categorizes into common characteristics and variable characteristics, we describe the difference product requirements with matrix (Figure 2). In Figure 2, a requirement (as PR_i) which emerges in all the products (as $Product_i$) is classified as commonality category. A requirement (as PR_{o1}) which only emerges in some products (as $Product_1$,$Product_3$,$Product_n$) is classified as variability category. The variability category includes *common, option, alternate, or* etc. *Or* characters do not show in Figure 2, because *or* is a subset of *option*.

Requirement	Characteristics	$Product_1$	$Product_2$	$Product_3$	·····	$Product_n$	
PR_{c1}	Common	√	√	√	√	√	Common Characters
···	Common	√	√	√	√	√	
PR_{cm}	Common	√	√	√	√	√	
PR_{o1}	Optional	√	×	√	×	√	Optional Characters
···	Optional	×	√	√	×	√	
PR_{om}	Optional	√	×	√	√	×	
PR_{a1}	Alternate	⊕	√	×	√	×	Alternate Characters
···	Alternate	√	×	⊕	×	⊕	
PR_{am}	Alternate	√	⊕	⊕	×	√	

√ : a product (as $Product_1$) possess a requirement(as PR_1)

× : a product (as $Product_2$) doesn't possess a requirement(as PRo_1)

⊕ : a requirement(as PRo_1) in a product (as $Product_2$) can be substituted with the other requirement

Figure 2 : Requirements matrix for SPL

The requirements matrix captures the basic requirements for the domain use case model. We explicitly depict the roles and use cases in domain use case model with the extended UML language. TABLE 1 represents the notation of the extended UML language in domain use case model.

TABLE 1 : Notation of the extended UML language

Concept	UML Construct	Stereotype
Common	Role or Use-Case	《Common》
Option	Role or Use-Case	《Option》
Alternate	Role or Use-Case	《Alternate》
Or	Role or Use-Case	《Or》
Optional Alternate	Role or Use-Case	《Optional Alternate》
Optional Or	Role or Use-Case	《Optional Or》

According to the product requirements matrix, we can construct the domain use-case model with the extended UML notation, and can revise the use-case model based on the requirements matrix and the reflected use case attributes.

The overlap between a basic requirement and the other basic requirement should be separate from the use-case model. Similarly, the overlap between a use case and the other use case should be separate from the use case model. If a use case contains an optional basic requirement, we will separate the optional basic requirement from the domain use case model by the extended UML language.

Dependency-constraint notation

A variability model can depict variation point, variant, variability dependency and variability constraint. The key of variability modeling is how to indicate the variability dependency with the use-case diagram, and combine effectively the variants according to the constraint rules, and then make the variability model accords with the requirement of application products.

The variation point and the variants which belong to the variation point are associated with variability dependency. One variant can associate with one or more variation points. Every variation point can include one variant at least or a set of variants. We build the application product according to the variability constraint while variants and variation point are tailored or bound. The variant must satisfy the variability dependencies and the variability constraint rules when the application product binds a variant. To be able to graphically represent the variability dependency-constraint defined under the variability model, we depict the variability dependency-constraint with the graphical notation for SPL in Figure 3, dependency denotes by the solid line, constraint denotes by the dotted line.

Variability modeling with extended UML

To implement the variability, we adopt the extended use-case model for variability modeling. Adding the roles and the use cases according to the product requirements matrix, and then respectively depict the semantics of constraint relationship among them by the commonality and variability. The whole process can be divided into four steps.

A. Establishing the initial use-case model. The roles or use case which emerges in every product will directly be extract to the use-case model. The role or use case which emerges in part products will be identified with variable role or variable use case.

B. Revising the use-case model. The use-case describes the variable roles or use cases on variation points, thus the use-case model include mandatory and variable use cases/roles.

C. Describing the variable use case scenes with variability dependency. The variability dependency includes variability type and associate. The variants of the variation point may be *mandatory, optional, alternative, or, optional alternative, optional or* etc., the associate includes generalization, inclusion, extend relationship between the variation point and the variants.

D. Depicting the variability constraint. In use case model, the variability constraints include variant to variant constraint, variation point to variant constraint, variation point to variation point constraint.

We describe the use case/role by TABLE 1 while constructing the use-case model with UML. If we will describe the variability type for use case/role, we need extend UML notation. TABLE 2 shows the variability category with the extended UML notation. In TABLE 2, assuming the variation point *Vp* is selected, we have the following definitions on its variants:

- Mandatory-The variant *v* must be bound to the variation point *Vp*.
- Optional-The variant *v* may or may not be bound to the variation point *Vp*.
- Alternative-Only one variant from a set of variants such as v_1 and v_2 can be bound to the variation point *Vp*.
- Or-One or more variants from a set of variants such as v_1 and v_2 can be bound to the variation point *Vp*.
- Optional Alternative-One variant from a set of alternative variants such as v_1 and v_2 may or may not be bound to the variation point *Vp*.
- Optional Or-One or more variants from a set of or variants such as v_1 and v_2 may or may not be bound to the variation point *Vp*.

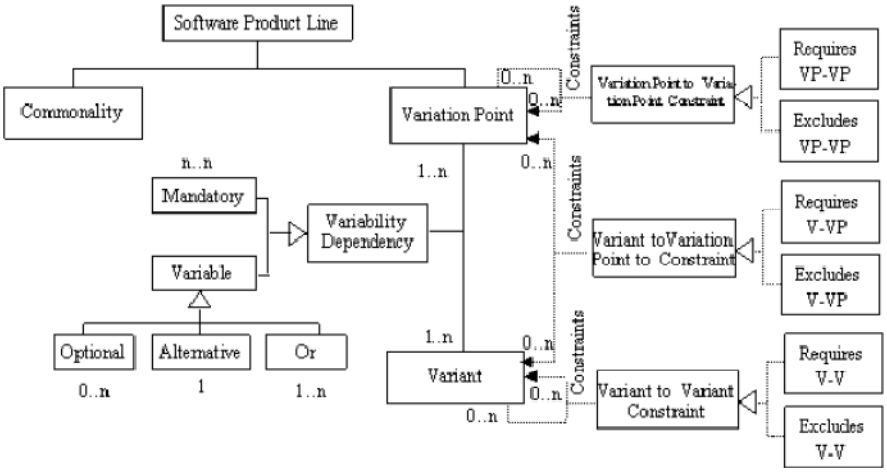

Figure 3 : The variability dependency-constraint notation for SPL

TABLE 2 : The notation of variability in the use-case diagram

Variability Type	Logic Representation	Cardinality	Extended UML Notation
Mandatory	$\mathcal{V}p \Leftrightarrow \mathcal{V}$	1	
Optional	$\mathcal{V} \Rightarrow \mathcal{V}p$	0,1	
Alternative	$\mathcal{V}p \Leftrightarrow (\mathcal{V}1 \oplus \mathcal{V}2)$	1	
Or	$\mathcal{V}p \Leftrightarrow (\mathcal{V}1 \vee \mathcal{V}2)$	1..n	
Optional Alternative	$(\mathcal{V}1 \oplus \mathcal{V}2) \Rightarrow \mathcal{V}p$	0..1	
Optional Or	$(\mathcal{V}1 \vee \mathcal{V}2) \Rightarrow \mathcal{V}p$	0..n	

Formalizing Constraint

Variability constraint mechanism is used to establish the constraint relationship in the use-case model between variant to variant, variation point to variant, variation point to variation point when variability is tailored or bounded in the use-case model. According to the rules of variability dependency and variability constraint, the variability model tailored which should satisfy the rules of variability dependency and variability constraint.

When establishing an application product model, it needs to tailor variability model of software product line, and decides whether to keep the variation point, but the model is still possible variable after tailoring. In order to ensure the model is correct after tailoring or binding, we use propositional logic to describe the rules of variability dependency and variability constraint.

Software product line variability model includes variation point, the variant, the variability dependency between variation point and variants, the variability constraint rules between variant to variant, variant to variation point, variation point to variation point.

Dependency Logic. When a variant is bound to variation point, *Vp* denotes variation point, *v* denotes variant, *T(Vp)* predicates that variation point is selected. $T(v_i)$ predicates that a variant v_i is selected in variation point *Vp*. *T(Vp)*, $T(v_i)$ value is:

$$T(Vp) = \begin{cases} \text{True} & Vp \text{ is selected} \\ \text{False} & Vp \text{ is not selected} \end{cases}$$

$$T(v_i) \begin{cases} \text{True} & v_i \text{ is selected when } Vp \text{ is selected} \\ \text{False} & v_i \text{ is not selected when } Vp \text{ is selected} \end{cases}$$

True *Vp* is selected

False *Vp* is not selected

True v_i is selected when *Vp* is selected

False v_i is not selected when *Vp* is selected

The four basic dependencies of *Mandatory, Option, Alternative, Or* can be defined as follow.

Binding variation point *Vp* must bind all the variants from a set of $v_1, v_2, \ldots, v_n$.

$$Mandatory(Vp, v_1, v_2, \ldots, v_n) = T(Vp) \Leftrightarrow (T(v_1) \wedge T(v_2) \wedge \cdots\cdots \wedge T(v_n))$$

The variants of $v_1, v_2, \ldots, v_n$ are bound to variation point *Vp* optionally.

$$Optional(Vp, v_1, v_2, \ldots, v_n) = T(v_1) \vee T(v_2) \vee \cdots\cdots \vee T(v_n)) \Rightarrow T(Vp)$$

Binding variation point *Vp* must bind only one variant from a set of $v_1, v_2, \ldots, v_n$.

$$Aternative(Vp, v_1, v_2, \ldots, v_n) = T(Vp) \Leftrightarrow (T(v_1) \oplus T(v_2) \oplus \ldots\ldots \oplus T(v_n))$$

Binding variation point *Vp* must bind one or more variants from a set of $v_1, v_2, \ldots, v_n$.

$$Or(Vp, v_1, v_2, \ldots, v_n) = T(Vp) \Leftrightarrow (T(v_1) \vee T(v_2) \vee \cdots\cdots \vee T(v_n))$$

Variant to Variant Constraint. There is a variant v_i which is bound to the variation point VP_k and belongs to its member, denotes as $B(VP_k, v_i)$. We can define as follow if two variants such as v_i and v_j that v_i requires v_j or v_i exclude v_j at the same variation point VP_k. Of course, v_i and v_j can not belong to the same variation point.

$$\text{Requires\_V\_V}(v_i, v_j) = B(VP_k, v_i) \rightarrow B(VP_k, v_j) \quad 1 \le i \le n,\ 1 \le j \le n,\ i \ne j$$

$$\text{Excludes\_V\_V}(v_i, v_j) = B(VP_k, v_i) \oplus B(VP_k, v_j) \quad 1 \le i \le n,\ 1 \le j \le n,\ i \ne j$$

*Variant to Variation Point Constraint. T*he variant-variation point constraint among a variant v_i of the variation point VP_k and the other variation point VP_m includes that v_i requires VP_m or v_i excludes VP_m. v_i requires VP_m means that the variant v_i requires the valid combinations of all variants which can be bound to the variation point VP_m, v_i excludes VP_m means that the variant v_i excludes all the valid combinations of all variants which can be bound to the variation point VP_m. We can define the propositional logic for variant to variation point constraint as follow.

$$\text{Requires\_V\_VP}(v_i, VP_m) = B(VP_k, v_i) \rightarrow \forall v_j\ (B(VP_m, v_j)) \quad 1 \le i \le n,\ 1 \le j \le n,\ i \ne j$$

$$\text{Excludes\_V\_VP}(v_i, VP_m) = B(VP_k, v_i) \rightarrow \neg(\forall v_j\ (B(VP_m, v_j))) \quad 1 \le i \le n,\ 1 \le j \le n,\ i \ne j$$

Variation Point to Variation Point Constraint. The constraint between the variation point VP_k and the variation point VP_m includes that VP_k requires VP_m or VP_k excludes VP_m. VP_k requires VP_m means that any variant of the variation point VP_k requires the valid combinations of all variants which can be bound to the variation point VP_m, VP_k excludes VP_m means that any variant of the variation point VP_k excludes all the valid combinations of all variants which can be bound to the variation point VP_m. We can define the propositional logic for variation point to variation point constraint as follow.

$$\text{Requires\_VP\_VP}(VP_k, VP_m) = \exists v_i\ (B(VP_k, v_i)) \rightarrow \forall v_j\ (B(VP_m, v_j)) \quad 1 \le i \le n,\ 1 \le j \le n,\ i \ne j$$

$$\text{Excludes\_VP\_VP}(VP_k, VP_m) = \exists v_i\ (B(VP_k, v_i) \rightarrow \neg(\forall v_j\ (B(VP_m, v_j))) \quad 1 \le i \le n,\ 1 \le j \le n,\ i \ne j$$

VARIABILITY MODEL OF MOBILE CASE STUDY

In mobile phone SPL, the commonality provides basic mobile functions, the variability supports the various mobile products by the special variants combination, then assembling the commonality and the variability to the new mobile phone application software. The part use-case diagram for variability mobile phone SPL is showed in Figure 4. We use the extended UML language to depict variability type, variability dependency and variability constraint. The mobile phone SPL(in short MP) includes the mandatory use case such as "Dial Mode", "Data Service", "Operating System" and "Transmission Mode", the optional use case such as "IM Software". "Operating System" use-case includes an alternative set of variants for "Android", "IOS", "Windows Phone" and Symbian. "Transmission Mode" use-case includes an alternative set of variants for "GPRS", "3G" and 4G. The "IM Software" use case includes an optional set of variants for "Wechat", "Fetion" and "UC". The "Dial Mode" use case includes a mandatory "PressKey Dial" and an optional "Voice Dial", the "Data Service" use case includes a mandatory "SMS" and an optional set of "Mobile Payment", "GPS" and "Navigation". The dependency is showed as follow.

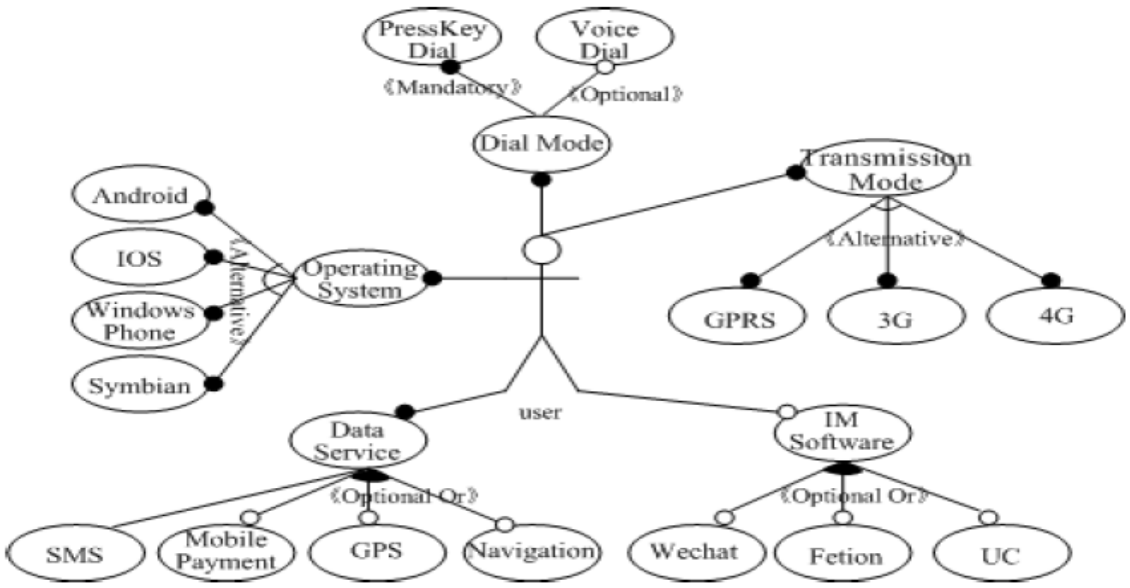

Figure 4 Variability Use-Case Diagram for Mobile Phone SPL

Mandatory(MP, Dial Model, Operating System, Data Service)=$T(MP)\Leftrightarrow(T(Dial\ Model)\wedge T(Operating\ System)\wedge T(Data\ Service))$

Optional(MP, Set Background, IM Software)=$(T(Set\ Background)\vee T(IM\ Software))\Rightarrow T(MP)$

Alternative(Operating System, Android, IOS, Windows Phone, Symbian)=$T(Operating\ System)\Leftrightarrow(T(Android)\oplus T(IOS)\oplus T(Windows\ Phone)\oplus T(Symbian))$

Alternative(Transmission, GPRS, 3G, 4G)=$T(Transmssion)\Leftrightarrow(T(GPRS)\oplus T(3G)\oplus T(4G))$

Mandatory(Dial Mode, PressKey Dial)=$T(Dial\ Mode)\Leftrightarrow T(PressKey\ Dial))$

Optional(Dial Mode, Voice Dial)=$T(VoiceDial)\Rightarrow T(Dial\ Mode)$

Optional(IM Software, Wechat, Fetion, UC)=$(T(Wetchat)\vee T(Fetion)\vee T(UC))\Rightarrow T(IM\ Software)$

Mandatory(Data Service, SMS) = $T(Data\ Service)\Leftrightarrow T(SMS)$

Optional(Data Service, Mobile Payment, GPS, Navigation)=$(T(Mobile\ Payment)\vee T(GPS)\vee T(Navigation))\Rightarrow T(Data\ Service)$

The constraint for the variant "Navigation" requires the variant "GPS" can be depicts as fellow.

Requires_V_V(Navigation, GPS)=$B(Data\ Service, Navigation)\rightarrow B(Data\ Service, GPS)$

The variant "Wechat" requires the variant "GPS" of the variation point "Data Service", at the same time it requires the variation point "Transmission Model", the constraints for variant "Wechat" can be depicts as fellow.

Requires_V_VP(Wechat, Transmission)∧Requires_V_V(Wechat, GPS)

$=(B(IM\ Software,\ Wechat)\rightarrow Alternative(Transmission\ Model,\ GPRS,\ 3G,\ 4G))\wedge(B(IM\ Software,\ Wechat)\rightarrow Mandatory(Data\ Service,\ GPS))$

=(B(IM Software, Wechat)→(B(Transmission Model, GPRS)⊕B(Transmission Model, 3G)⊕B(Transmission Model, 4G))∧(B(IM Software, Wechat)→B(Data Service, GPS))

CONCLUSIONS AND FUTURE WORK

Variability modeling allows a developer to compose variants in different combinations tailoring the design to specific requirements, to reuse variants in different application products. In our Mobile Phone case study, we propose an approach of variability modeling with extended UML language base on the SPL variability analysis, the use case requirement-matrix, mutual dependencies and constraints. The variability modeling approach not only supports the variable type such as mandatory, optional, alternative, but also supports the constraint modeling for SPL variability. Finally, the variability modeling of Mobile Phone case study verifies the effectiveness of the proposed method.

Our future work will continually investigate the formalization representation for feature modeling, and solve the potential conflicts among variants, variation points.

ACKNOWLEDGMENT

The work was funded by the research project of Chongqing Nature Science Foundation (cstc2013 jcyjA40066) and the research project of Chongqing University of Arts and Sciences Foundation(Y2013JX50).

REFERENCES

[1] Klaus Pohl, Günter Böckle, Frank van Der Linden; Software product line engineering-foundations[M], Principles, and Techniques, Springer-Verlag Berlin Heidelberg, **(2005)**.

[2] H.Gomaa; Designing software product lines with UML[C], IEEE software engineering workshop, IEEE Computer Society, 160-216 **(2005)**.

[3] K.Nie, L.Zhang, Z.Geng; Product line variability modeling based on model difference and Merge[C], Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual, IEEE, 509-513 **(2012)**.

[4] S.H.Ripon; A unified tabular method for modeling variants of software product line[J], ACM SIGSOFT Software Engineering Notes, **37(3)**, 1-7 **(2012)**.

[5] S.Ripon, K.Azad, S.J.Hossain et al.; Modeling and analysis of product-line variants[C], Proceedings of the 16th International Software Product Line Conference-Volume 2, ACM, 26-31 **(2012)**.

[6] R.Michel, A.Classen, A.Hubaux et al.; A formal semantics for feature cardinalities in feature diagrams[C], Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems, ACM, 82-89 **(2011)**.

[7] A.Classen, Q.Boucher, P.Heymans; A text-based approach to feature modelling: Syntax and semantics of TVL[J], Science of Computer Programming, **76(12)**, 1130-1143 **(2011)**.

[8] K.Czarnecki, P.Grünbacher, R.Rabiser et al.; Cool features and tough decisions, A comparison of variability modeling approaches[C], Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems, ACM, 173-182 **(2012)**.

[9] A.Haber, H.Rendel, B.Rumpe et al.; Hierarchical variability modeling for software architectures[C], Software Product Line Conference (SPLC), 2011 15th International, IEEE, 150-159 **(2011)**.

[10] Zou Shengheng, Zhang wei, Zhao haiyan; Modeling variability in software product family, Journal of Software, **16(1)**, 37- 49 **(2005)**.

[11] K.Lee, K.C.Kang; Feature dependency analysis for product line component design[M], Software Reuse: Methods, Techniques, and Tools, Springer Berlin Heidelberg, 69-85 **(2004)**.

[12] Nunes, Vinicius; Variability management of reliability models in software product lines, An expressiveness and scalability analysis, Proceedings-2012 6th Brazilian Symposium on Software Components, Architectures and Reuse, SBCARS, 51-60 **(2012)**.

[13] Simmonds, Jocelyn; Variability in software process models, Requirements for adoption in industrial settings, 2013 4th International Workshop on Product Line Approaches in Software Engineering, Please Proceedings, 33-36 **(2013)**.

[14] Peng, Xin, Liu, Jindu, Zhao, Wenyun; Towards feature-oriented variability reconfiguration in dynamic software product lines, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v 6727 LNCS, 52-68.

[15] Kulesza, Uirá, Oliveira, A.Edson; Modeling variabilities from software process lines with compositional and annotative techniques, A quantitative study, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v7983 LNCS, 153-168.

[16] Khosravi,Ramtin; Modeling variability in business process models using UML, Proceedings-International Conference on Information Technology, New Generations, ITNG, 82-87 **(2008)**.