Knowledge Based Method to Validate Feature Models

Abdelrahman O. Elfaki, Somnuk Phon-Amnuaisuk, Chin Kuan Ho

Center of Artificial Intelligent and Intelligent computing,

Multimedia University, Cyberjaya,

Malaysia

abdelrahman.osman.06@mmu.edu.my,

somnuk.amnuaisuk@mmu.edu.my, ckho@mmu.edu.my

Topics

- Introduction
- Problem statements
- Extending Feature Models by Variability Notations
- Variation points and variants as a predicates
- Dependency Constraint Rules
- Operations on the Automated Analysis of Feature Models
- Discussions and Comparison with Previous Work
- Conclusion and future work

Introduction

- To say what are you presenting

Terminology

- Features are requirement specifications of a software product.
- A feature model is used to describe a software product-line by describing commonalities and differences among software products to be produced from the software product-line.
- A single product line may generate many software products constrained by features in the feature model.

Terminology

- *Variability*

Problem Statements

- Features are requirement specifications of a software product.

- What can be improved?

Proposed methodology

- Features are requirement specifications of a software product.

- What can be improved?

BNF

Dependency Constraint Rules

1. $\forall x, y: type(x, variant) \wedge type(y, variant) \wedge require\_v\_v(x, y) \wedge select(x) \Longrightarrow select(y)$
2. $\forall x, y: type(x, variant) \wedge type(y, variant) \wedge exclude\_v\_v(x, y) \wedge select(x) \Longrightarrow notselect(y)$
3. $\forall x, y: type(x, variant) \wedge type(y, variationpoint) \wedge require\_v\_vp(x, y) \wedge select(x) \Longrightarrow select(y)$

$\forall x, y: type(x, variant) \wedge type(y, variationpoint) \wedge require\_v\_vp(x, y) \wedge select(y) \Longrightarrow select(x)$

Dependency Constraint Rules Cont….

4. $\forall x, y$: $type(x, variant) \land type(y, variationpoint) \land excludev\_vp(x, y) \land select(x) \Rightarrow notselect(y)$

$\forall x, y$: $type(x, variant) \land type(y, variationpoint) \land exclude\_v\_vp(x, y) \land select(y) \Rightarrow notselect(x)$

5. $\forall x, y$: $type(x, variationpoint) \land type(y, variationpoint) \land require\_vp\_vp(x, y) \land select(x) \Rightarrow select(y)$

Dependency Constraint Rules Cont….

6. $\forall x, y$: *type(x, variationpoint)* $\wedge$ *type(y, variationpoint)* $\wedge$ *exclude_vp_vp(x, y)* $\wedge$ *select(x)* $\Rightarrow$ *notselect(y)*

7. $\forall x, y$: *type(x, variant)* $\wedge$ *type(y, variationpoint)* $\wedge$ *select(x)* $\wedge$ *variants(y, x)* $\Rightarrow$ *select(y)*

8. $\exists x \; \forall y$:*type(x, variant)* $\wedge$ *type(y, variationpoint)* $\wedge$ *select(y)* $\wedge$ *variants(y, x)* $\Rightarrow$ *select(x)*

9. $\forall x, y$: *type(x, variant)* $\wedge$ *type(y, variationpoint)* $\wedge$ *notselect(y)* $\wedge$*variants(y, x)* $\Rightarrow$ *notselect(x)*

Variation points and variants as a predicates

- type: Describes the type of feature; variation point or variant. *e.g.* : type (view-type, variationpoint) , type (register,variant) .

- variant: Identifies the variant of specific variation point. *e.g.*: variant (view-type, notregister)

- Max : Identifies the maximum number allowed to be selecting of specific variation point. *e.g.* max (payment-by, 4)

Extending Feature Models by Variability Notations

- requires-v-v : variant requires variant
- excludes-v-v : variant excludes variant
- requires-v-vp : variant requires variation point
- excludes-v-vp : variant excludes variation point
- requires-vp-vp : variation point requires variation point
- excludes-vp-vp : variation point excludes variation point

Variation points and variants as a predicates Cont….

- min: Identifies the minimum number allowed to be selecting of specific variation point. *e.g.* min (payment-by, 1) .

- common : describe the commonality of specific feature. *e.g.* common (search-name, yes). If the feature is not common, the second slot in the predicate will become No -as example- common (register, no).

Dependency Constraint Rules Cont….

10. $\forall x, y: type(x, variant) \wedge type(y, variationpoint) \wedge common(x) \wedge variants(y, x) \wedge select(y) \Longrightarrow select(x)$

11. $\forall y: type(y, variationpoint) \wedge common(y) \Longrightarrow select(y)$

12. $\forall x, y: type(x, variant) \wedge type(y, variationpoint) \wedge variants(y, x) \wedge select(x) \Longrightarrow sum(y,(x)) \leq max(y,z)$

13. $\forall x, y: type(x, variant) \wedge type(y, variationpoint) \wedge variants(y, x) \wedge select(x) \Longrightarrow sum(y,(x)) \geq min(y,z)$

Dependency Constraint Rules Cont….

- From these rules we can define a full common variant, variant included in any product, as:

$$\forall x,y{:}type(x,variant) \wedge type(y,variationpoint) \wedge variants(y,x) \wedge common(y) \wedge common(x) \Longrightarrow full\_common(x)$$

Operations on the Automated Analysis of Feature Models

1. Propagation

Selection of variant *n*, *select(n)*, is propagated from selection of variant *x*, *select(x)*, in three cases:

$\forall x,y,z,n{:}type(x,variant)\wedge variants(y,x)\wedge select(x)\wedge requires\_vp\_vp(y,z)\wedge type(n,variant)\wedge variant(z,n)\wedge common(n)\Longrightarrow select(n).$

$\forall x,n{:}\ type(x,variant)\wedge type(n,variant)\wedge select(x)\wedge requires\_v\_v(x,n)\Longrightarrow select(n).$

$\forall x,z,n{:}type(x,variant)\wedge select(x)\wedge type(z,variationpoint)\wedge requires\_v\_vp(x,z)\wedge type(n,variant)\wedge variants(z,n)\wedge common(n)\Longrightarrow select(n).$

Operations on the Automated Analysis of Feature Models Cont…

2. Explanation

Selection of variant *n*, *select (n)*, fails due to selection of variant *x*, *select(x)*, in three cases:

$\forall x,y,n: type(x,variant) \wedge select(x) \wedge type(y,variationpoint) \wedge variants(y,x) \wedge type(n,variant) \wedge excludes\_v\_vp(n,y) \Rightarrow notselect(n).$

$\forall x,y,z,n: type(x,variant) \wedge select(x) \wedge type(y,variationpoint) \wedge variants(y,x) \wedge variants(z,n)\ excludes\_vp\_vp(y,z) \Rightarrow notselect(n).$

$\forall x,n\ type(x,variant) \wedge select(x) \wedge type(n,variant) \wedge excludes\_v\_v(x,n) \Rightarrow notselect(n).$

Operations on the Automated Analysis of Feature Models Cont…

- In addition to defining the source of error, these rules can be used to prevent the errors. The predicate *notselect(n)* validate users by prevent selection.

Operations on the Automated Analysis of Feature Models Cont…

- Optimization

This operation returns the output according to a given function or predefined criteria.

Optimization function can be added as a fact

e.g. price(variant , num). And can handel extra function.

Operations on the Automated Analysis of Feature Models Cont…

4. Dead Feature Detection

A variant *x* can be a dead feature in 3 cases:

$\forall x,y,z,n: type(x,variant) \wedge type(y,variationpoint) \wedge variants(y,x) \wedge type(z,variant) \wedge type(n,variationpoint) \wedge variants(n,z) \wedge common(n,yes) \wedge common(z,yes) \wedge excludes\_v\_vp(z,y) \Longrightarrow dead\_feature(x).$

$\forall x,y,z: type(x,variant) \wedge type(y,variationpoint) \wedge variants(y,x) \wedge type(z,variationpoint) \wedge common(z,yes) \wedge excludes\_vp\_vp(z,y) \Longrightarrow dead\_feature(x).$

$\forall x,y,n: type(n,variant) \wedge type(y,variationpoint) \wedge variants(y,n) \wedge common(y,yes) \wedge common(n,yes) \wedge type(x,variant) \wedge excludes\_v\_v(n,x) \Longrightarrow dead\_feature(x).$

Operations on the Automated Analysis of Feature Models Cont…

5. Inconsistency

The inconsistency (error) can be detected in five cases:

$\forall x,y{:}type(x,variant) \land type(y,variant) \land requires\_v\_v(x,y) \land excludes\_v\_v(y,x) \Longrightarrow error.$

$\forall x,y{:}type(x,variationpoint) \land type(y,variationpoint) \land requires\_vp\_vp(x,y) \land excludes\_vp\_vp(y,x) \Longrightarrow error.$

$\forall x,y,n,z{:}\ type(x,variant) \land type(y,variationpoint) \land variants(y,x) \land type(n,variant) \land type(z,variationpoint) \land variants(z,n) \land requires\_v\_v(x,n) \land excludes\_vp\_vp(y,z) \Longrightarrow error.$

Operations on the Automated Analysis of Feature Models Cont…

$\forall x,y,z: type(x,variant) \wedge common(x,yes) \wedge type(y,variationpoint) \wedge variants(y,x) \wedge type(z,variationpoint) \wedge excludes\_v\_vp(x,z) \wedge requires\_vp\_vp(y,z) \Longrightarrow error.$

$\forall x,y,z: type(x,variant) \wedge common(x,yes) \wedge type(y,variationpoint) \wedge variants(y,x) \wedge type(z,variationpoint) \wedge requires\_v\_vp(x,z) \wedge excludes\_vp\_vp(y,z) \Longrightarrow error.$

Feature models can contain some other complicated forms of inconsistencies like (A requires B) and (B requires C) and (C excludes A). To avoid this complication the following rules were defined:

Operations on the Automated Analysis of Feature Models Cont…

$\forall x,y,z: type(x,variant) \wedge type(y,variant) \wedge$
$requires\_v\_v(x,y) \wedge type(z,variant) \wedge requires\_v\_v(y,z) \Rightarrow requires\_v\_v(x,z).$

$\forall x,y,z: type(x,variationpoint) \wedge type(y,variationpoint) \wedge requires\_vp\_vp(x,y) \wedge type(z,variationpoint) \wedge requires\_vp\_vp(y,z) \Rightarrow requires\_vp\_vp(x,z).$

$\forall x,y,z: type(x,variant) \wedge type(y,variationpoint) \wedge requires\_v\_vp(x,y) \wedge type(z,variationpoint) \wedge requires\_vp\_vp(y,z) \Rightarrow requires\_v\_vp(x,z).$

$\forall x,y,z: type(x,variant) \wedge type(y,variant) \wedge requires\_v\_v(x,y) \wedge type(z,variationpoint) \wedge$
$requires\_v\_vp(y,z) \Rightarrow requires\_v\_vp(x,z).$

Operations on the Automated Analysis of Feature Models Cont…

$\forall x,y,z: type(x,variant) \land type(y,variant) \land excludes\_v\_vp(x,y) \land type(z,variant) \land excludes\_v\_v(y,z) \Rightarrow excludes\_v\_v(x,z).$

$\forall x,y,z: type(x,variationpoint) \land type(y,variationpoint) \land excludes\_vp\_vp(x,y) \land type(z,variationpoint) \land excludes\_vp\_vp(y,z) \Rightarrow excludes\_vp\_vp(x,z).$

$\forall x,y,z: type(x,variant) \land type(y,variationpoint) \land excludes\_v\_vp(x,y) \land type(z,variationpoint) \land excludes\_vp\_vp(y,z) \Rightarrow excludes\_v\_vp(x,z).$

$\forall x,y,z: type(x,variant) \land type(y,variant) \land excludes\_v\_v(x,y) \land type(z,variationpoint) \land excludes\_v\_vp(y,z) \Rightarrow excludes\_v\_vp(x,z).$

Operations on the Automated Analysis of Feature Models Cont…

- Using backtracking mechanism the above rules can solve more complex shapes of inconsistency such as ((A requires B) and (B requires C) and (C requires D) and (D requires F) and (F excludes A)).
- Our proposed method to auto detected error in feature model is better than that suggested by Trinidad (P. Trinidad 2008) because it defines two types of error in feature models.

Operations on the Automated Analysis of Feature Models Cont…

6. Cardinality

min(variation pint name, num).

max(variation point name, num).

Discussions and Comparison with Previous Work

- Automated consistency check among constraints during modeling
- Our proposed method can substantially define and provide an automated mechanism to the following operations:
- Definition of propagation states
- Provide explanation: define source of error + guide user to solve it.
- Detect dead features
- Detect inconsistency

Conclusion

- By modeling variability using knowledge base rules we can get both formalized variability specifications, and support selection process within variability more precisely.

- The proposed knowledge base can be used to configure new product from SPL, analyzing SPL and produce error free feature model

Future Work

- Extend our work using constraint handling rules (CHR) to calculate and obtain all products, calculate variability (the ratio between all product and all variants in the model) , and calculate commonality.
- Develop software tool to support our method.
- Validate our method by applying it to real life case studies from industry.