Master Informatique - Université de Montpellier - Ingénierie logicielle - 2017

Lignes de produits logiciels
Création, édition et analyse de modèles de caractéristiques

Objectif : prise en main de `FAMILIAR` et `SPLOT`, deux outils permettant de manipuler des lignes de produits au travers de modèles de caractéristiques.

1 Modèles de caractéristiques

Les modèles de caractéristiques (*feature models*) sont une famille de langages de description visuels qui permettent de décrire un ensemble de caractéristiques ainsi que des dépendances entre ces caractéristiques. Ils sont utilisés pour représenter un ensemble de produits appartenant à une même famille, de manière compacte et compréhensible.

Un modèle de caractéristiques organise de manière hiérarchique un ensemble de caractéristiques dans un arbre : la caractéristique racine est la plus générale (elle représente souvent le nom de la famille modélisée) alors que les caractéristiques les plus basses dans la hiérarchie sont les plus spécialisées. Une arête de l'arbre représente donc une relation de raffinement entre une caractéristique mère et une (plusieurs) caractéristique(s) fille(s). Un produit correspond à un ensemble de caractéristiques sélectionnées dans cet arbre. Ces arêtes peuvent être décorées pour contraindre la sélection. Si on sélectionne une caractéristique qui possède une sous-caractéristique : une arête *optionnelle* définit que la sous-caractéristique n'est pas obligatoirement sélectionnée, et une arête *obligatoire* force la sous-caractéristique à être sélectionnée. On trouve aussi des contraintes sur les groupes de caractéristiques. Si on sélectionne une caractéristique qui possède un groupe de sous-caractéristiques : dans un groupe *or*, au moins une des sous-caractéristiques doit être sélectionnée, alors que dans un groupe *xor*, exactement une doit être sélectionnée. Les arêtes correspondant à ces 4 relations sont exposées dans la Figure 1 (gauche). Des contraintes qui ne peuvent être exprimées dans la hiérarchie (implications *cross-tree*, exclusions) peuvent être rajoutées textuellement.

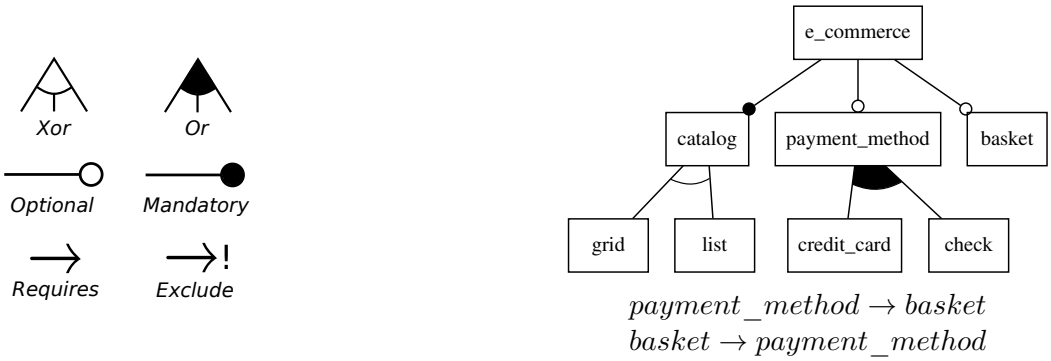

FIGURE 1 – Exemple de modèle de caractéristiques (droite) et ses différentes relations (gauche)

La Figure 1 (droite) représente un exemple de modèle de caractéristiques sur une famille d'applications de e-commerce. On peut lire que : une application de e-commerce possède obligatoirement un catalogue ; ce catalogue peut être affiché sous forme de grille ou de liste, mais pas les deux ; l'application peut optionnellement proposer des méthodes de paiement ; elle peut proposer un paiement par carte, par chèque, ou bien

les deux ; une application de e-commerce peut éventuellement proposer une gestion de panier ; si la gestion de panier est sélectionnée, l'application doit posséder des méthodes de paiement, et inversement.

La configuration $\{e\_commerce, catalog, grid, basket, payment\_method, credit\_card\}$ respecte l'ensemble des contraintes du modèle, et représente donc une variante possible de la famille de produits. Cependant, la configuration $\{e\_commerce, catalog, grid, basket\}$ ne représente pas une variante possible, car la contrainte $basket \rightarrow payment\_method$ n'est pas respectée. La liste des 8 configurations valides du modèle de la Figure 1 sont listées dans la Table 1.

TABLE 1 – Liste des 8 configurations décrites par le modèle de caractéristiques de la Fig. 1

	e_commerce	catalog	grid	list	paym_method	credit_card	check	basket
c1	x	x	x					
c2	x	x		x				
c3	x	x	x		x	x		x
c4	x	x	x		x		x	x
c5	x	x	x		x	x	x	x
c6	x	x		x	x	x		x
c7	x	x		x	x		x	x
c8	x	x		x	x	x	x	x

Les modèles de caractéristiques sont le standard *de facto* pour modéliser et représenter les connaissances liées à la variabilité d'une famille de produits. Ils permettent une meilleure gestion des éléments communs d'un ensemble de variantes, et des éléments spécifiques de certaines variantes, favorisant ainsi la réutilisation : les coûts et temps de développement sont alors réduits, et la qualité des produits ainsi obtenus est meilleure.

2 FAMILIAR

`FAMILIAR` (pour *FeAture Model scrIpt Language for manIpulation and Automatic Reasoning*) est un *domain-specific language* pour la manipulation de modèles de caractéristiques. Il permet entre autre de créer des modèles de caractéristiques conformes à ceux décrits en Section 1, de réaliser des opérations de modification (édition, fusion de modèles) ainsi que des opérations d'analyse telles que la vérification de la validité du modèle, la détection des caractéristiques inutilisées, la génération des configurations possibles, la comparaison de modèles, etc.

2.1 Téléchargement et installation

Le langage `FAMILIAR` s'accompagne d'un environnement disponible en deux versions : une version basique utilisable en console, et une version graphique. Ces deux applications sont téléchargeables sous la forme de fichiers JAR :

Version basique :
`http://mathieuacher.com/pub/FAMILIAR/releases/FML-basic-1.1.jar`

Version graphique :

`http://mathieuacher.com/pub/FAMILIAR/releases/FML-environment-1.1.jar`

Pour utiliser l'environnement, lancez simplement l'une des commandes suivantes dans un terminal :
— pour la version basic : `java -jar FML-basic-1.1.jar`
— pour la version graphique : `java -jar FML-environment-1.1.jar`

La documentation de `FAMILIAR` peut être trouvée ici :
`https://github.com/FAMILIAR-project/familiar-documentation/tree/master/manual`

2.2 Syntaxe

Pour construire un nouveau modèle de caractéristiques, il faut utiliser le constructeur "`FM`" :

```
fm1 = FM (A : B C [D]; B: (E|F) ; D : (J|K)+ ; (!C | D) ; )
```

— A est la caractéristique **racine**
— B, C et D sont les caractéristiques filles de A : B et C sont **obligatoires**, D est **optionnelle**
— E et F forment un **groupe XOR**, et sont les caractéristiques filles de B
— J et K forment un **groupe OR**, et sont les caractéristiques filles de D
— (!C | D) est l'équivalent de (C $\rightarrow$ D), une implication *cross-tree*
— Une contrainte d'exclusion entre C et D (C $\rightarrow$!D) s'écrirait donc (!C | !D)

Les détails de la notation textuelle interne sont disponibles au lien suivant :
`https://github.com/FAMILIAR-project/familiar-documentation/blob/master/manual/featuremodel.md`

2.3 Environnement basique

Lancez l'environnement basique sur un terminal. Vous devez obtenir une console propre à FAMILIAR.

Question 1 : Spécifiez le modèle de caractéristiques de la Figure 1 dans `FAMILIAR` en utilisant la syntaxe textuelle vue précédemment.

Question 2 : Vérifiez que votre spécification est correcte :
— Comptez le nombre de configurations décrites par votre modèle avec l'opération `count nomFM`. Vous devez obtenir le nombre 8.
— Listez l'ensemble des configurations valides du modèle avec l'opération `configs nomFM`. Correspondent-elles aux 8 configurations de la Table 1 ?

Question 3 : Utilisez l'opération `cores nomFM` de `FAMILIAR` sur votre modèle. Que fait cette opération ? Vérifiez votre théorie en vous basant sur les configurations listées dans la Table 1.

2.4 Environnement graphique

Lancez l'environnement graphique sur un terminal. Vous devez voir apparaître une fenêtre dans laquelle un modèle de caractéristiques *wiki* est déjà défini. Notez aussi la présence d'une console comme celle utilisée avec l'environnement basique.

Question 4 : Créez un nouveau modèle de caractéristiques (`File > new FAMILIAR FM`). Vous pouvez renseigner le nom de votre modèle (qui apparaîtra dans l'onglet du haut) et le nom de la caractéristique racine.

Question 5 : En faisant un clic droit sur la racine, vous pouvez renommer votre modèle (`Rename FM name`). Changez le nom actuel pour *MdC1*. Avec l'option `Rename Feature`, vous pouvez changer le nom de la caractéristique sélectionnée : renommez la en *e_ commerce*.

Question 6 : Spécifiez le modèle de caractéristiques de la Figure 1 graphiquement en utilisant `New Feature Child`. Des contraintes *cross-tree* peuvent être ajoutées en utilisant `New Constraint`.

Question 7 : Comptez le nombre de configurations, et affichez-les en utilisant les opérations `Count Valid Configs` et `Valid Configs` dans le menu `Reasoning`. Ces opérations sont toujours exécutables depuis la console : testez-les.

On considère à présent un second modèle de caractéristiques, toujours sur des applications de e-commerce :

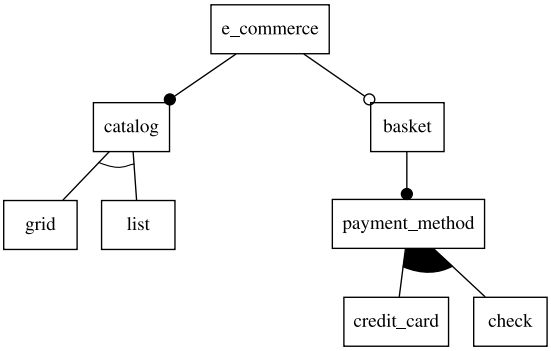

FIGURE 2 – Autre exemple de modèle de caractéristiques sur les applications de e-commerce

Question 8 : Dans `FAMILIAR`, créez un nouveau modèle de caractéristiques appelé *MdC2*, représentant le modèle de la Figure 2.

Question 9 : Comparez *MdC1* et *MdC2* avec l'opération `Compare FMs` du menu `Reasoning`. Que signifie le résultat obtenu ? Vérifiez en listant les configurations valides des deux modèles.

Question 10 : Spécifiez le modèle de la Figure 3 et énumérez son ensemble de configurations. Que peut-on dire sur les caractéristiques C et F ? Que peut-on dire sur la caractéristique E ? Utilisez les opérations `dead` et `falseOptionals` pour vérifier votre théorie.

Question 11 : Modifier le modèle de caractéristiques afin de corriger ces "anomalies", sans changer son ensemble de configurations valides.

3 Mise en situation

Une entreprise vous contacte afin que vous réalisiez un modèle de ligne de produits pour leurs nouveaux mailers. Voici leurs exigences :

"*Nous souhaitons proposer des clients mails personnalisés pour chacun de nos clients. Chaque mailer peut gérer une boite de réception, qui peut soit être divisée en plusieurs sous-dossiers, soit contenir tous les mails*

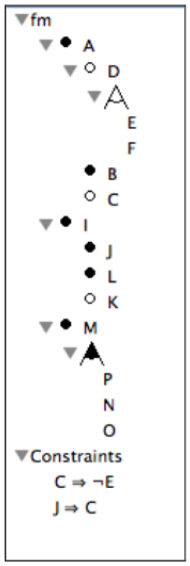

FIGURE 3 – Modèle de caractéristiques

dans un seul dossier. Un mailer peut éventuellement proposer d'étiqueter les mails avec des labels. Les labels sont soit définis par défaut (personnel, travail, loisir), soit entièrement personnalisables. Dans le deuxième cas, un gestionnaire de personnalisation de labels doit être implémenté. Si la boite de reception est divisée en plusieurs sous-dossiers, la gestion des étiquettes est obligatoire. Une option de compactage des mails peut aussi être proposée. Le compactage peut être automatique, manuel, ou bien les deux."

Question 12 : Identifiez les caractéristiques qui vous semblent pertinentes et réalisez un modèle de caractéristiques avec `FAMILIAR` pour représenter cette ligne de produits. Réalisez des opérations d'analyse sur votre modèle final pour vérifier qu'il ne contient pas d'anomalies (validité, caractéristiques inutilisées, faux optionels ...). L'entreprise vous demande combien de mailers différents respectant leurs exigences vont pouvoir être créés : utilisez les opérations de `FAMILIAR` afin de répondre à cette question.

Question 13 : Faites sur papier un diagramme de classes UML respectant les caractéristiques de votre modèle de caractéristiques. Choisissez une configuration valide de votre modèle, et faites le diagramme d'instance correspondant à cette configuration. Toutes les classes sont-elles utilisées dans cette instanciation ?

Question 14 : Implémenter ces classes en Java. Chaque classe contiendra (au moins) une méthode `toString()` qui renverra une chaîne de caractères contenant le nom de la classe, et qui fera appel au `toString()` des instances avec lesquelles elle est en relation.

Question 15 : Il est temps de générer le code de votre mailer automatiquement. Faites un `main()` qui créée une instance de mailer en fonction d'une configuration donnée. Affichez le `toString()` de cette instance : affiche-t-elle bien les classes correspondant à la configuration donnée ?

4 SPLOT

`SPLOT` (pour *Software Product Line Online Tools*) est un site qui repertorie plus de 800 modèles de caractéristiques et qui propose aux utilisateurs des outils en ligne pour créer de nouveaux modèles, les éditer, les analyser ainsi que de les configurer. Il se trouve à l'adresse suivante :

`http://www.splot-research.org/`

Question 13 : Allez dans l'onglet *Automated Analysis* et sélectionnez dans la liste le modèle nommé `SPL_e_commerce`. Une fois sélectionné, cliquez sur le bouton "click here" en haut de la page. Vous retrouvez des statistiques sur le modèle de la Figure 1 dans la première table. Lancez les analyses automatiques sur la deuxième et la troisième table (`click to run`) et étudiez les résultats. Sont-ils en accord avec ceux obtenus précédemment avec `FAMILIAR` ?

Question 14 : À présent, allez dans l'onglet *Product Configurator*, puis cliquez sur le lien *Click to use the Interactive Configuration feature*. Choisissez le modèle `SPL_e_commerce` précédent et lancez le configurateur. Cette interface permet de réaliser une dérivation de produit à partir du modèle sélectionné. Vous voyez apparaître le modèle de caractéristiques sur la gauche. L'avancement de la dérivation est illustrée sur la droite. Que se passe-t-il lorsque l'on sélectionne la caractéristique *grid* ? Pourquoi ? Mêmes questions pour la caractéristique *payment_ method*.