Université De Montpellier
Faculté Des Sciences

Niveau : Master 2

Module : IA pour le Génie Logiciel

HAI916I

Travaux de construction de Feature Models avec FCA

Supervisé par :
M. Marianne Huchard

Réalisé par :
Allouch Yanis
Blanchard Guilhèm
Bonneaud Maël
Kaci Ahmed

2021/2022

Table des matières

Introduction

Dans ce rapport, nous allons établir un raisonnement permettant la construction d'un Feature Model (FM) en se basant sur l'extraction de variabilité reposant sur l'analyse formelle de concept. Et ce, en axant nos réponses autour des questions fournies dans le cours.

Nous allons, dans un premier temps, exposer une méthode que nous avons déduit en se basant sur la construction d'un Feature Model établit à partir du treillis et de l'ontologie nommés *Brigthness*. Dans un second temps, nous allons utiliser cette méthodologie pour réaliser un feature model en s'appuyant sur un autre treillis et une autre ontologie intitulés *TextSize*.

1 Feature Model Brightness

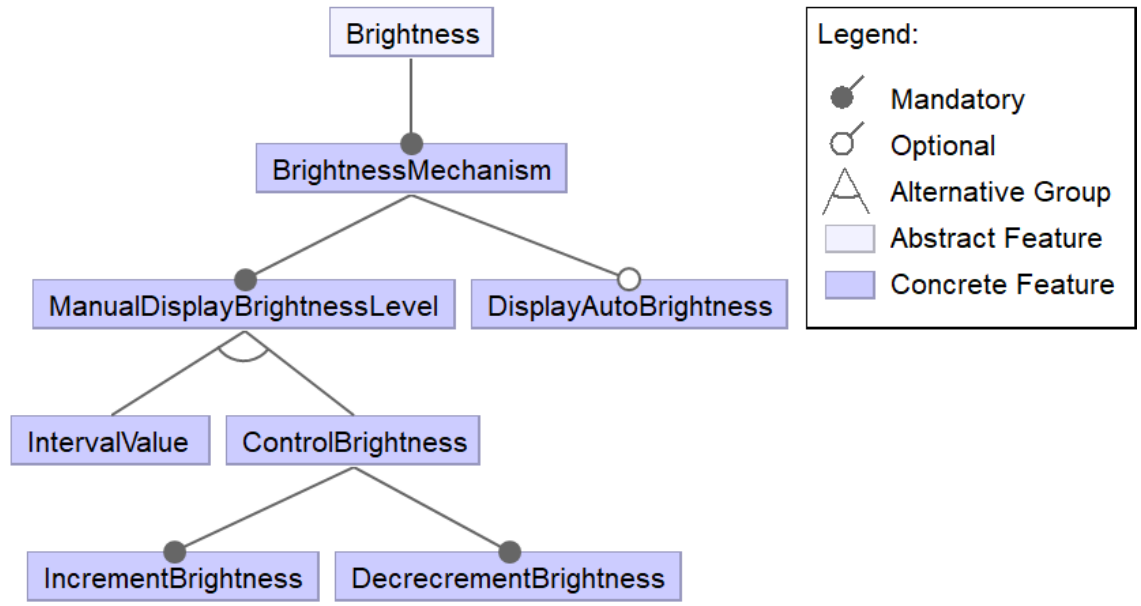

FIGURE 1 – Un Feature model construit par les règles décrites ci-dessous

1.1 Dans quel ordre avons-nous procédé

1. La première chose que nous avons faîte, c'est de partir du treillis. On démarre de la racine du treillis pour nommer la racine du feature model d'après le nom du concept racine.
2. On remarque que dans le concept courant, il y a deux attributs. Tous les attributs du concept racine du treillis, seront alors relié avec une relation obligatoire (*mandatory*) a la racine du feature model.
3. Il faut vérifier en utilisant l'ontologie, si les relations *mandatory* décrite sur le feature model sont compatibles avec la hiérarchie proposée dans l'ontologie. Dans le cas contraire, il faut transposer la hiérarchie de l'ontologie sur le feature model. Exemple :
   — D'après l'ontologie ManualDisplayBrightnessLevel est "**une sorte de**" BrightnessMechanism. Il est donc plus logique de placer ManualBrightnessLevel en dessous de BrightnessMechanism.

4. Lors de la transposition de la hiérarchie du feature model nous pouvons avoir des cas où il serait plus judicieux de regrouper des features par un concept abstrait avec une relation *mandatory* ou *optional*.
5. Pour construire le reste du feature model, il faut à partir du concept courant, créer les feature en fonction des attributs des concepts fils (direct).
6. Pour continuer cette construction du feature model nous déterminons les features à créer en se basant sur les attributs des concepts fils d'un concept donné.
7. On peut relier une feature crée avec une autre feature la spécialisant si cette dernière existe.

Dans ce qui suit nous détaillerons la méthode suivie, pour déterminer la nature (obligatoire, optionnelle, Or et Xor) des relations choisies entre les features.

1.2 Avons-nous plutôt utilisé l'ontologie ou le treillis

Arcs du feature model

Comme nous l'avons détaillé dans la question précédente, nous avons utilisé le treillis et l'ontologie pour déterminer les arcs du feature model.

Indications

— Optionnel :
  — Si dans le treillis, il existe un branchement d'au moins deux concepts introducteur d'attributs, alors, on peut considérer ces attributs ou les attributs d'un sous concept de ces deux concepts comme des features optionnelles. Et en se basant sur la hiérarchie fournit dans l'ontologie.
  exemple :
  Dans le feature model de Brightness nous avons déduit une relation optionnelle entre *BrightnessMechanism* et *DisplayAutoBrightness*

— Obligatoire :
  — Si dans le treillis, il existe un concept introducteur d'attributs reliant une disjonction de concepts, alors, on peut considérer ces attributs comme des features obligatoires.
  — Si dans le treillis, il existe un concept introducteur de plusieurs attributs, alors on peut considérer ces attributs comme des features obligatoires si et seulement si, d'après l'ontologie, ces mêmes attributs ont un même super-attribut (parent) ou si c'est la racine (réflexivité).
  Exemple :
  Nous avons une relation obligatoire entre *BrightnessMechanism* et *ManualDisplayBrightnessLevel*.

— Xor :

— Si dans le treillis, il existe un branchement de plusieurs sous concepts avec un concept père et que dans l'ontologie les attributs des sous concepts ont le même attribut parent avec lequel ils sont relié alors on déduit une relation XOR.

— Or :

— Si dans le treillis, il existe un branchement de plusieurs sous concepts avec un concept père et qu'on n'a pas déduit de relation XOR dans l'étape précédente sur ces attributs alors on déduit une relation OR, étant donné qu'il existe une disjonction non-exclusive.

Exemple Figure 2.

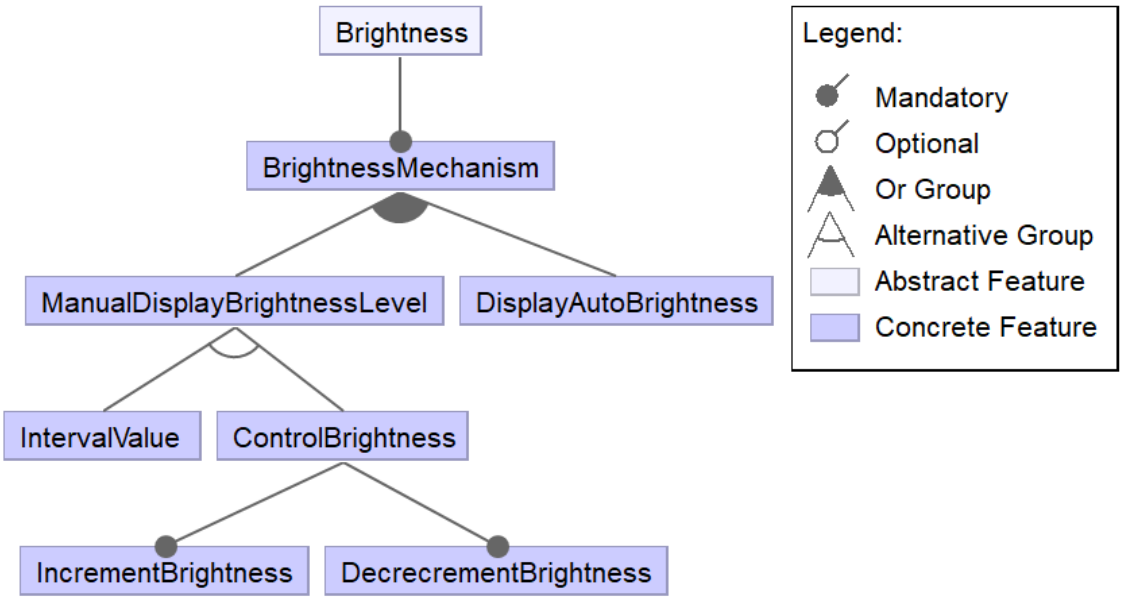

FIGURE 2 – Un Feature Model du contexte formel Brightness

Contraintes textuelles

Si dans le treillis on a un concept possédant un attribut B qui hérite d'un concept possédant un attribut A et que dans l'ontologie l'attribut B ne spécialise pas l'attribut A (pas de relation de descendance hiérarchique) alors on crée une contrainte d'implication entre ces deux attributs.

1.3 Informations non utilisées de l'ontologie

L'ontologie était principalement utilisée afin d'avoir un rapprochement entre nos déductions du treillis et la sémantique. Cependant, il peut y avoir des informations de l'ontologie que nous n'avons pas repérés lors de la réalisation de nos Feature Model. En effet, il existe plusieurs Feature Model pour modéliser un même domaine

métier. Alors avec une interprétation du treillis et de l'ontologie différente, un feature model différent auraient pu voir le jour.

1.4 Informations non utilisées du treillis

Dans notre analyse du treillis et de l'ontologie, nous avons essayé d'utiliser le plus d'informations possible. Cependant, il se peut qu'il y a éventuellement des informations que nous n'avons pas repérés dans le treillis, sachant que l'ontologie nous a complété un grand nombre d'informations.

1.5 Ajout des features (des termes non présents dans l'ontologie)

Oui, nous avons ajouté une feature (qui aurait pu être abstraite), nommé *Control-Brightness*.

1.6 Feature ignoré (des termes présents dans l'ontologie que vous n'avez pas utilisé)

Non, aucune.

1.7 Informations qui nous ont manqué pour effectuer certains choix

Oui on aurait aimé avoir le contexte formel détaillé.

1.8 Remarque

Aucune remarque.

2 Feature Model TextSize

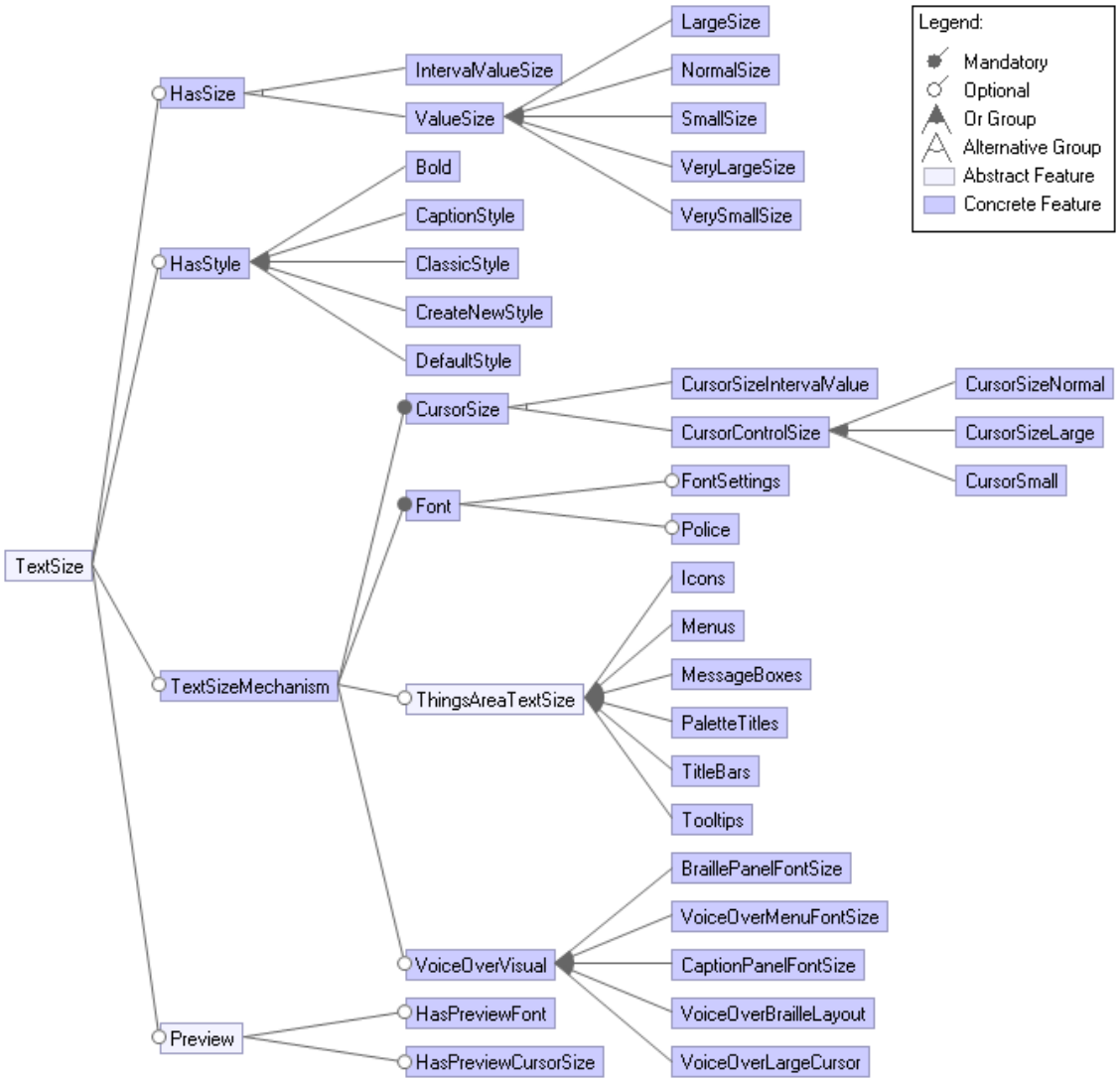

FIGURE 3 – Un Feature Model du Concept Formelle associé a *TextSize*

2.1 Dans quel ordre avons-nous procédé

Nous avons suivi la méthodologie précédemment décrite en sous-section 1.1.

2.2 Avons-nous plutôt utilisé l'ontologie ou le treillis

Comme nous l'avons détaillé dans la question précédente, nous avons utilisé le treillis et l'ontologie pour déterminer les arcs du feature model.

2.3 Ajout des features (des termes non présents dans l'ontologie)

Oui, nous avons ajouté deux features au modèle (qui aurait pu être abstraite), nommé *ValueSize* et *CursorControlSize*.

2.4 Feature ignoré (des termes présents dans l'ontologie que vous n'avez pas utilisé)

Nous avons dû ignorer, certaines features. Parce que nous n'avons pas trouvé de justifications suffisantes à leur ajout dans le FM. Les features ignorer sont, *ShowExtendedTextStyle* et *ShowTextStyle*.

2.5 Informations qui nous ont manqué pour effectuer certains choix

Oui on aurait aimé avoir le contexte formel détaillé.

2.6 Remarque

Nous avons observé des erreurs syntaxiques, probablement du a la re-copie manuelle des attributs. Nous avons aussi relevé des attributs manquants.

Voici les erreurs que nous avons relevées dans l'ontologie :

— Syntaxique (écrit autrement dans le treillis) :
  1. VoiceOverBrailleLayout,
  2. VoiceOverVisual.

— Manquant (absent dans le treillis) :
  1. Preview,
  2. ShowTextStyle,
  3. ShowExtendedTextStyle.

Voici les erreurs que nous avons relevées dans le treillis :

— Syntaxique (écrit autrement dans l'ontologie) :
  1. HasSize,
  2. HasStyle.

— Manquant (absent dans l'ontologie) :
  1. HasIntervalValue,
  2. HasPreviewFont,
  3. HasPreviewCursorSize.

3 Conclusion

En conclusion, nous avons détaillé notre approche, en axant nos réponses autour des questions fournies dans le cours. Nous avons complété notre approche, par quelques images illustrant nos exemples de *feature model*, obtenus en minant les treillis et ontologies à disposition sur le Moodle.

Nous pouvons retenir que les deux sources d'information, ontologie et treillis, apportent des informations pertinentes pour la réalisation d'un feature model. Le treillis permet de connaître l'ensemble des features, leurs caractères obligatoire ou non et leurs liens de parenté.

Références bibliographiques

[1] R Al-Msie'deen et al. « Reverse engineering feature models from software configurations using formal concept analysis ». In : *Proceedings of the eleventh international conference on concept lattices and their applications, Košice, Slovakia*. Citeseer. 2014, p. 95-106.

[2] Klaus Pohl, Günter Böckle et Frank Van Der Linden. *Software product line engineering : foundations, principles, and techniques*. T. 1. Springer, 2005.