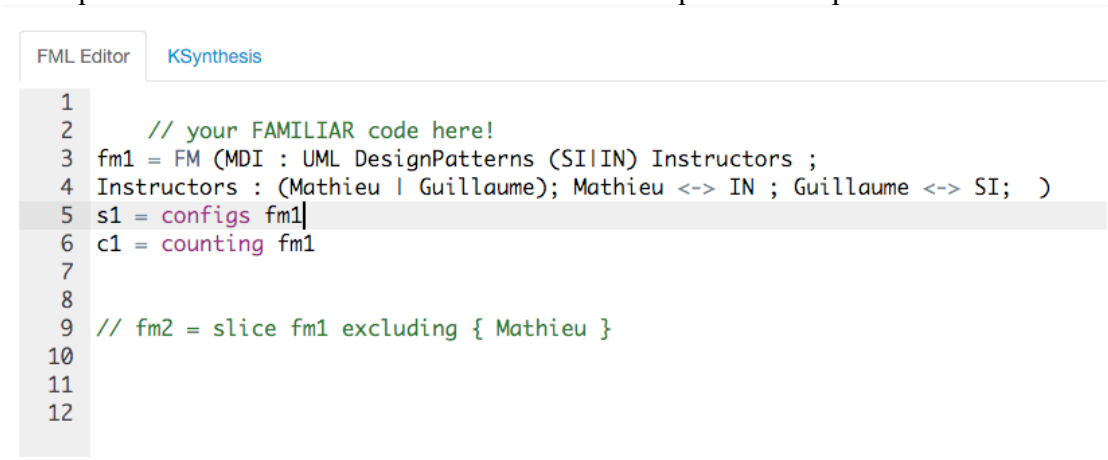
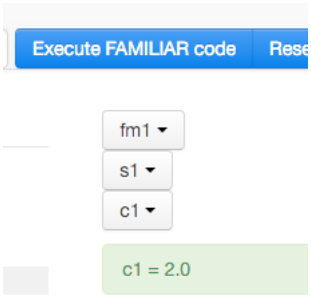
TD/TP : Ligne de produits logiciels, feature model

Le préambule, Exercice 1 et Exercice 2 sont réalisés par le professeur avec des explications interactives. Exercice 3 et Exercice 4 sont à réaliser par les étudiants.

Accéder à la page Web :
http://familiar.variability.io/ide/familiar

Vous pouvez commencer le travail et exécuter votre premier script :





Les « opérations » de FAMILIAR (configs, cores, counting, isValid, etc.) sont documentées ici:
https://github.com/FAMILIAR-project/familiar-documentation/blob/master/manual/

Exercice 1. Syntaxe et sémantique des feature models (avec un outil)

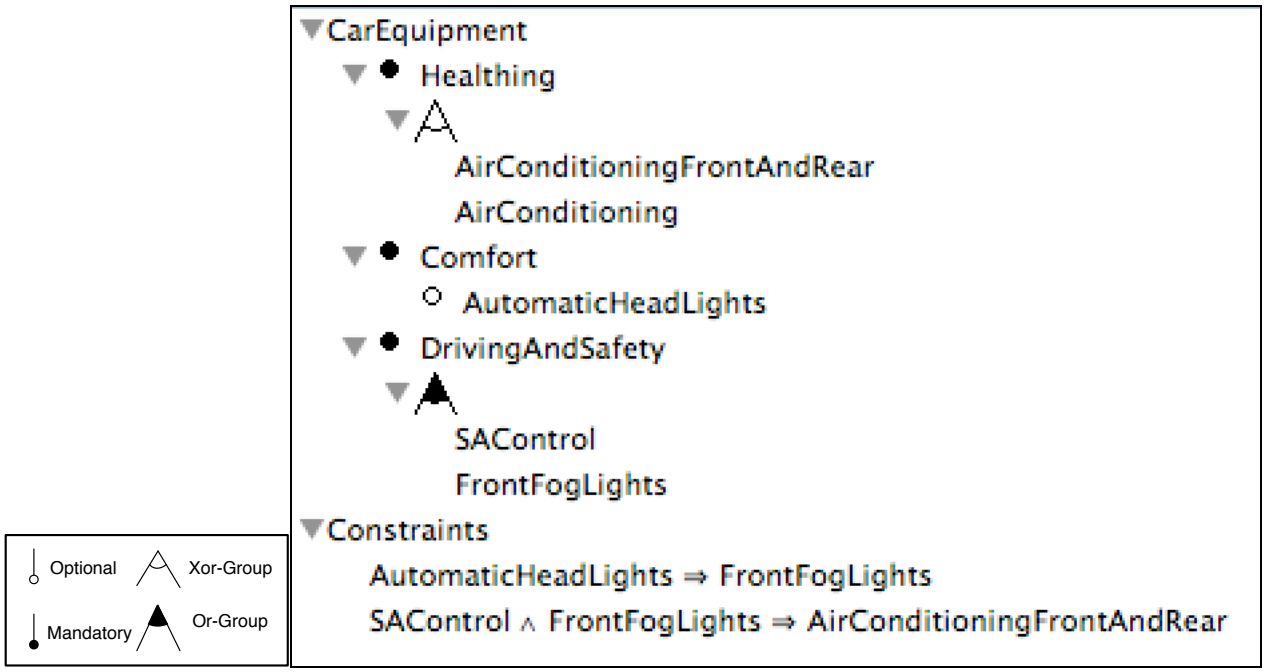


Figure 1.

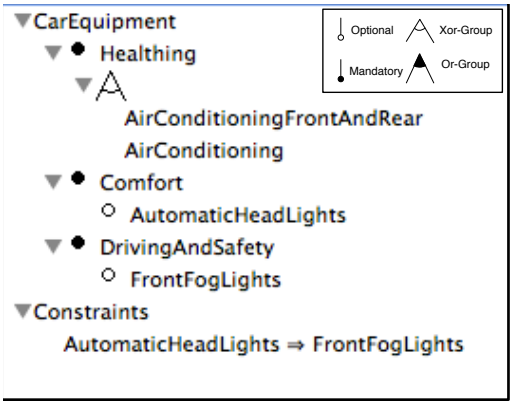
Question #0: Spécifier le feature model de la Figure 1 avec FAMILIAR, en utilisant la notation interne textuelle : https://github.com/FAMILIAR-project/familiar-documentation/blob/master/manual/featuremodel.md

Internal notation

Building a new feature model uses the constructor `FM`:

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

- A is the **root**
- B, C and D are child-features of A: B and C are **mandatory** whereas D is **optional**
- E and F form a **XOR-group** and are child-features of B
- G, H and I form an **optional XOR-group** and are child-features of C
- J and K form an **OR-group** and are child-features of D
- (!C | D) is equivalent to (C -> D) and is an **internal constraint** of the feature model



{CarEquipment, Comfort,
DrivingAndSafety,
Healthing}

{AirConditioning, FrontFogLights}
{AutomaticHeadLights, AirConditioning, FrontFogLights}
{AutomaticHeadLights, FrontFogLights, AirConditioningFrontAndRear}
{AirConditioningFrontAndRear}
{AirConditioning}
{AirConditioningFrontAndRear, FrontFogLights}

```
fmCarEquipment = FM (CarEquipment : Healthing DrivingAndSafety Comfort ; // 3 mandatory features
                                    Healthing : (AirConditioning|AirConditioningFrontAndRear) ; // Xor
                                    DrivingAndSafety : [FrontFogLights] ; // optional
                                    Comfort : [AutomaticHeadLights] ; // optional
                                    // cross-tree constraints
                                    AutomaticHeadLights -> FrontFogLights ; )
```

```
fml> co = cores fmCarEquipment
co: (SET) {CarEquipment;Healthing;DrivingAndSafety;Comfort}
fml> fmCarEquipment.*
res6: (SET) {DrivingAndSafety;AirConditioningFrontAndRear;Comfort;Healthing;FrontFogLights;AirConditioning;AutomaticHeadLights;CarEquipment}
fml> setDiff fmCarEquipment.* co
res7: (SET) {AutomaticHeadLights;FrontFogLights;AirConditioning;AirConditioningFrontAndRear}
fml>
res1: (DOUBLE) 6.0
```

6

Question #1: Vérifier que votre spécification textuelle est conforme à la Figure 1 en produisant une énumération exhaustive de l'ensemble des configurations valides avec l'opération « configs »

Question #2: Quelles sont les features qui sont incluses dans n'importe quelle configuration? Utiliser l'opération « cores »

Question #4: Proposer un feature model avec une hiérarchie différente mais caractérisant le même ensemble de configurations. Vérifier le résultat en utilisant l'opération « compare »

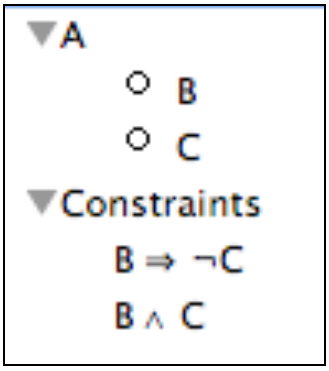


Figure 2

Question #5: Spécifier le feature model de la Figure 2 avec FAMILIAR. Vérifier que le nombre de configurations valides du feature model de la Figure 2 est 0 en utilisant l'opération « counting » et « isValid ».

Question #6: Que se passe-t-il si on relâche la première contrainte? Que se passe-t-il si on relâche la deuxième contrainte? Adresser la question avec les opérations de FAMILIAR.

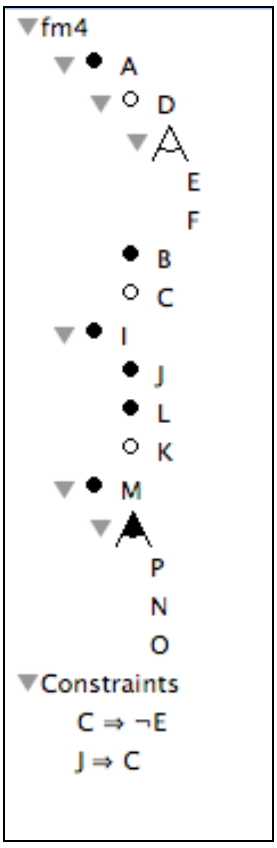


Figure 3

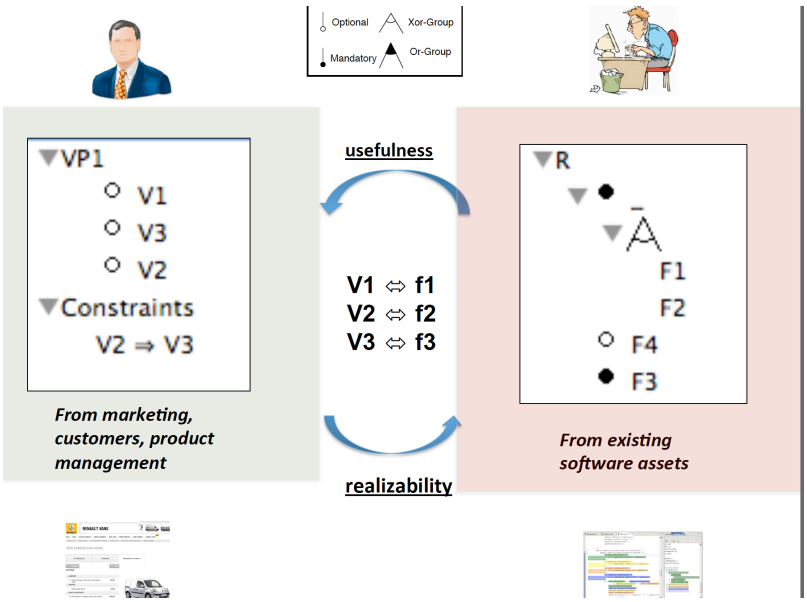
Question #7: Spécifier le feature model de la Figure 3 avec FAMILIAR.
Enumérez l'ensemble des configurations valides.
Que peut-on dire sur les features C et F ?
Que peut-on dire sur la feature E ?

Utiliser les opérations « deads » et « falseOptionals » pour vérifier vos dires.

Question #8: Corrigez les « anomalies », i.e., réécrire le feature model de manière à ce qu'il exprime le même ensemble de configuration mais cette fois-ci les informations de variabilité sont en adéquation avec les configurations valides du feature model.
Vérifier avec FAMILIAR que les anomalies sont bien corrigées, i.e., qu'elles ne sont plus présentes.

Question #9: Que se passe-t-il si on relâche la première contrainte? Que se passe-t-il si on relâche la deuxième contrainte? Réitérez la série de questions précédentes avec FAMILIAR pour chaque suppression de contrainte.

Exercice 2. Consistance et réécriture de vues



Une organisation veut proposer à ses clients des options de configuration (features) et décident de modéliser les configurations autorisées avec le feature model de gauche.

Question #10: Faîtes une énumération exhaustive de l'ensemble des configurations valides du feature model de gauche avec FAMILIAR.

Cette même organisation a une plateforme logicielle dans laquelle il y a de la variabilité – une feature est optionnelle et il y a deux alternatives d'implémentation. Le feature model de droite représente cette variabilité logicielle.

Question #11: Faîtes une énumération exhaustive de l'ensemble des configurations valides du feature model de droite avec FAMILIAR.

L'idée de l'organisation est, qu'étant donné une configuration choisie par un client (et valide par rapport au feature model de gauche), il y a au moins une configuration logicielle correspondante (et valide par rapport au feature model de droite).
Ainsi le client peut configurer son produit sans aborder les détails techniques de la plateforme logicielle.

Il y a une correspondance entre le feature model de gauche et le feature model de droite sous la forme de contraintes.

Question #12: Faîtes une énumération exhaustive de l'ensemble des configurations valides du feature model de gauche et de droite une fois que la correspondance entre les deux a été établie via les contraintes. Que peut-on en conclure ?

Exercice 3. Des matrices aux feature models

Nous considérons les 2 matrices de comparaison ci-dessous (extraites de https://en.wikipedia.org/wiki/Comparison_of_video_converters)

Overview [edit]

Video converter	Developer	Licensing scheme	Supported platform			Website
			Windows	Mac OS X	Linux	
Any Video Converter	AVCLabs	Freeware	Yes	Yes	No	any-video-converter.com
Audials Tunebite 10 Platinum	Audials	Shareware	Yes	Yes	No	audials.com
Avidemux	Mean,Gruntster,Fahr	Free and open-source	Yes	Yes	Yes	www.avidemux.org
Dr. DivX	DivX, Inc.	Free and open-source	Yes	Yes	No	labs.divx.com/DrDivX
DVDVideoSoft Free Studio	DVDVideoSoft	Freeware (ad supported)	Yes	No	No	dvdvideosoft.com
FFmpeg	FFmpeg project	Free and open-source	Yes	Yes	Yes	ffmpeg.org
FormatFactory	Chen Jun Hao	Freeware (ad supported)	Yes	No	No	formatoz.com
Freemake Video Converter	Freemake	Freeware (ad supported)	Yes	No	No	freemake.com
HandBrake	Handbrake Project	Free and open-source	Yes	Yes	Yes	handbrake.fr
MediaCoder	Stanley Huang	Freeware (ad supported)	Yes	No	No	www.mediacoderhq.com
MEncoder	The MPlayer Project	Free and open-source	Yes	Yes	Yes	mplayerhq.hu
OggConvert	Tristan Brindle	Free and open-source	Experimental	No	Yes	oggconvert.tristanb.net
Prism Video Converter	NCH Software	Freeware (and paid version)	Yes	Yes	No	nchsoftware.com/prism/
SUPER	eRightSoft	Freeware (ad supported)	Yes	No	No	www.erightsoft.com
Transcode	Transcode Team	Free and open-source	No	No	Yes	transcoding.org
VirtualDub	Avery Lee	Free and open-source	Yes	No	No	virtualdubmod.sourceforge.net
XMedia Recode	Sebastian Dörfler	Freeware	Yes	No	No	www.xmedia-recode.de

Input [edit]

	Supported input container formats														
Video converter	3GP	AVI	Blu-ray video	DVD video	FLV	Matroska	MP4	MPEG-PS	Ogg	QuickTime	SVCD	TS	TOD	VCD	WMV
Any Video Converter	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Audials Tunebite 10 Platinum	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Avidemux	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
FFmpeg	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Freemake Video Converter	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
FormatFactory	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DVDVideoSoft Free Studio	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
HandBrake	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SUPER[1]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Prism Video Converter	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes
XMedia Recode	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Question #13 : L'objectif est de produire un feature model qui correspond à ces deux matrices de produit (donc on veut un seul feature model). A toute configuration valide de ce feature model correspondra au moins un produit. Les « features » seront donc issues des les 2 matrices.
Avant d'élaborer le feature model, il faut s'interroger (1) sur la pertinence de certaines caractéristiques (e.g., Website) ; (2) la manière de structurer l'information sous forme de hiérarchie. Attention également car certains produits ne sont que dans une matrice (en première approximation, on s'intéressera uniquement aux produits qui sont dans les deux matrices)

Exercice 4. JHipster

« Jhipster is a tool for generating a complete and modern Web stack, at the back and front-end level (server side with Spring Boot, client-side with AngularJS and Bootstrap, and Yeoman, Bower, Grunt and Maven for builiding the application).
Jhipster is quite popular with more than 6000 stars in Github. It is available online : https://jhipster.github.io/ »

1. Nous voulons modéliser les configurations autorisées par Jhipster. Expliquer pourquoi jouer avec le configurateur et essayer toutes les combinaisons possibles d'options n'est pas une solution viable.
2. Ecrire un feature model qui caractérise l'ensemble des configurations du configurateur Jhipster. Une configuration valide doit être autorisée par le configurateur. A cause des limites d'une stratégie manuelle basée sur l'observation du configurateur, il est nécessaire d'étudier *l'implémentation* du configurateur

https://github.com/jhipster/generator-jhipster/blob/master/generators/server/prompts.js
https://github.com/jhipster/generator-jhipster/blob/master/generators/client/prompts.js
https://github.com/jhipster/generator-jhipster/blob/master/generators/app/prompts.js

3. Calculer le nombre valide configurations de Jhipster
4. Comparons nos solutions
5. Expliquer comment on pourrait « tester » des configurations de Jhipster à partir du feature model (et potentiellement trouver des configurations défectueuses)

Le travail sur Jhipster est à rendre pour le 30 septembre 2017 en uplodant votre solution dans le dossier « JhipsterFMs » ici : http://tinyurl.com/jhipster-MIAGE-RE1718
Vous pouvez rendre le travail à deux.