Software Product Line Engineering

modeling and managing variability of software intensive systems

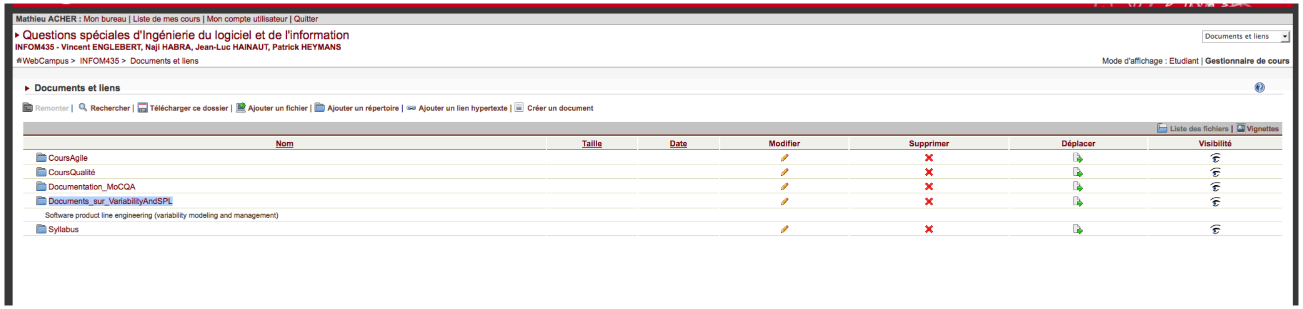
Dr. Mathieu Acher
email: macher@fundp.ac.be

Prof. Patrick Heymans

University of Namur
PReCISE Research Centre

Material

- http://www.fundp.ac.be/etudes/cours/page_view/INFOM435/
  – Folder: “Documents_sur_VariabilityAndSPL”
  – Slides, exercises, evaluation



Mathieu ACHER : Mon bureau | Liste de mes cours | Mon compte utilisateur | Quitter

Questions spéciales d'Ingénierie du logiciel et de l'information
INFOM435 - Vincent ENGLEBERT, Naji HABRA, Jean-Luc HAINAUT, Patrick HEYMANS

WebCampus > INFOM435 > Documents et liens

Mode d'affichage : Etudiant | Gestionnaire de cours

Documents et liens

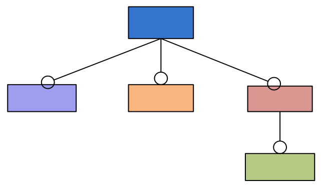
Nom	Taille	Date	Modifier	Supprimer	Déplacer	Visibilité
CoursAgile						
CoursQualité						
Documentation_MoCQA						
Documents_sur_VariabilityAndSPL						
Software product line engineering (variability modeling and management)						
Syllabus						

Documents et liens
Remonter | Rechercher | Télécharger ce dossier | Ajouter un fichier | Ajouter un répertoire | Ajouter un lien hypertexte | Créer un document
Liste des fichiers | Vignettes

- Email: macher@fundp.ac.be

Previously

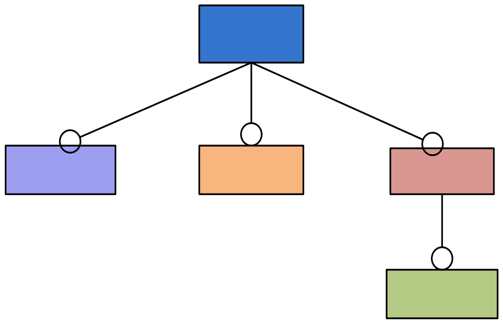
- **Variability Modeling**
  - the importance of modeling variability
  - the formalism of feature models
  - feature models + other artefacts
- **Practice**
  - Existing car configurators
  - TVL (language)



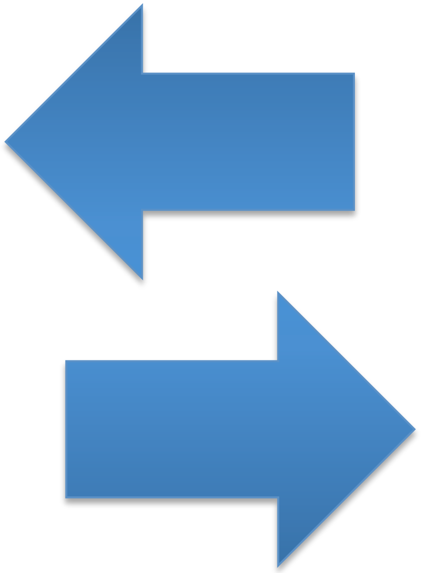
Feature Model

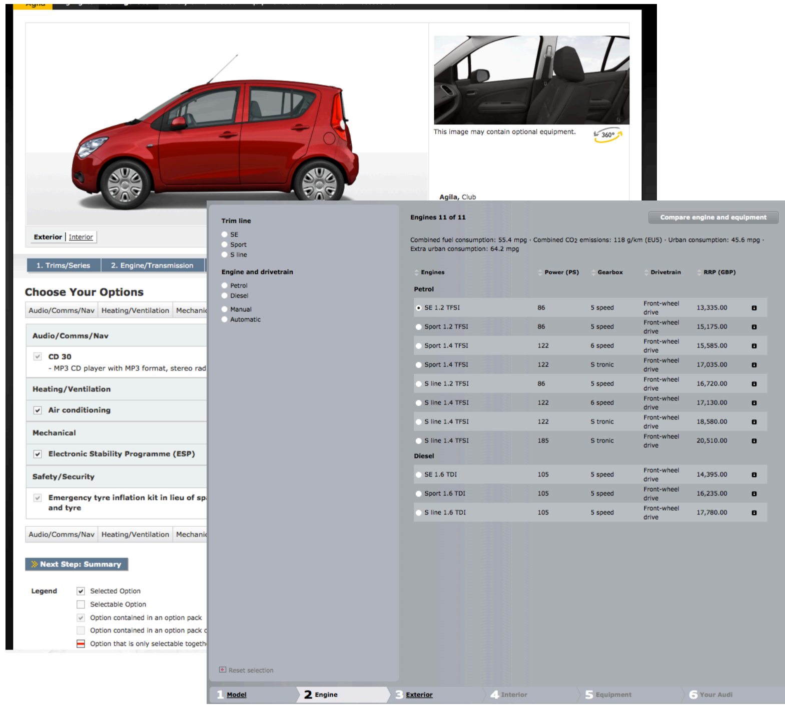
Running project

- Re-engineer a car configurator
  - we are making some progress, right?



Variability Model (Feature Model)





This image may contain optional equipment.

Agila, Club

Exterior | Interior

1. Trims/Series | 2. Engine/Transmission

Choose Your Options

Audio/Comms/Nav | Heating/Ventilation | Mechanic

Audio/Comms/Nav

✓ **CD 30**
- MP3 CD player with MP3 format, stereo rad

Heating/Ventilation

✓ **Air conditioning**

Mechanical

✓ **Electronic Stability Programme (ESP)**

Safety/Security

✓ **Emergency tyre inflation kit in lieu of sp and tyre**

Audio/Comms/Nav | Heating/Ventilation | Mechanic

Next Step: Summary

Legend	
✓	Selected Option
☐	Selectable Option
✓	Option contained in an option pack
☐	Option contained in an option pack c
🟥	Option that is only selectable togeth

Trim line
- SE
- Sport
- S line

Engine and drivetrain
- Petrol
- Diesel
- Manual
- Automatic

Reset sele

Engines 11 of 11

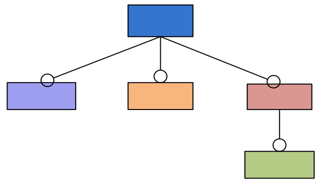
Compare engine and equipment

Combined fuel consumption: 55.4 mpg · Combined CO_2 emissions: 118 g/km (EU5) · Urban consumption: 45.6 mpg · Extra urban consumption: 64.2 mpg

Engines	Power (PS)	Gearbox	Drivetrain	RRP (GBP)
Petrol				
● SE 1.2 TFSI	86	5 speed	Front-wheel drive	13,335.00
Sport 1.2 TFSI	86	5 speed	Front-wheel drive	15,175.00
Sport 1.4 TFSI	122	6 speed	Front-wheel drive	15,585.00
Sport 1.4 TFSI	122	S tronic	Front-wheel drive	17,035.00
S line 1.2 TFSI	86	5 speed	Front-wheel drive	16,720.00
S line 1.4 TFSI	122	6 speed	Front-wheel drive	17,130.00
S line 1.4 TFSI	122	S tronic	Front-wheel drive	18,580.00
S line 1.4 TFSI	185	S tronic	Front-wheel drive	20,510.00
Diesel				
SE 1.6 TDI	105	5 speed	Front-wheel drive	14,395.00
Sport 1.6 TDI	105	5 speed	Front-wheel drive	16,235.00
S line 1.6 TDI	105	5 speed	Front-wheel drive	17,780.00

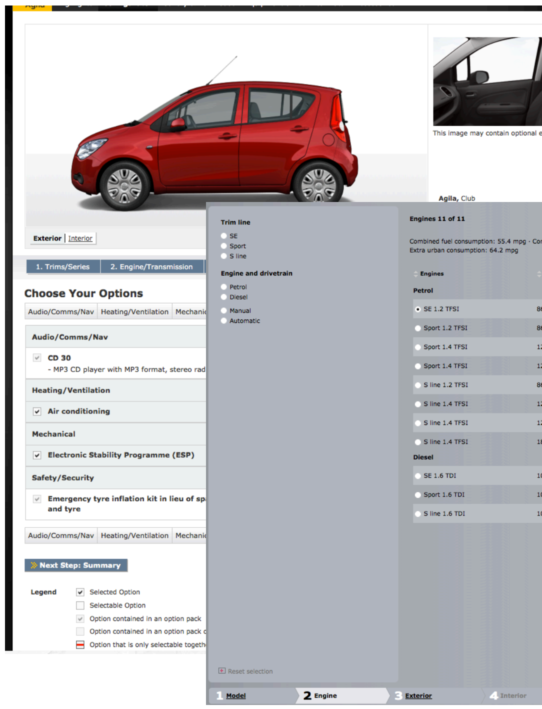
Previously

Language (TVL, FAMILIAR)



Feature Model

Automated reasoning



Exterior | Interior

This image may contain optional e

Agila, Club

1. Trims/Series | 2. Engine/Transmission

Choose Your Options

Audio/Comms/Nav | Heating/Ventilation | Mechanic

Audio/Comms/Nav

✓ **CD 30**
- MP3 CD player with MP3 format, stereo rad

Heating/Ventilation

✓ **Air conditioning**

Mechanical

✓ **Electronic Stability Programme (ESP)**

Safety/Security

✓ **Emergency tyre inflation kit in lieu of sp and tyre**

Audio/Comms/Nav | Heating/Ventilation | Mechanic

Next Step: Summary

Legend	
✓	Selected Option
☐	Selectable Option
✓	Option contained in an option pack
☐	Option contained in an option pack c
▭	Option that is only selectable togeth

Trim line
- ○ SE
- ○ Sport
- ○ S line

Engine and drivetrain
- ○ Petrol
- ○ Diesel
- ○ Manual
- ○ Automatic

Reset selection

Engines 11 of 11

Combined fuel consumption: 55.4 mpg · Co
Extra urban consumption: 64.2 mpg

Engines

Petrol
- ● SE 1.2 TFSI 8
- ○ Sport 1.2 TFSI 8
- ○ Sport 1.4 TFSI 1
- ○ Sport 1.4 TFSI 1
- ○ S line 1.2 TFSI 8
- ○ S line 1.4 TFSI 1
- ○ S line 1.4 TFSI 1
- ○ S line 1.4 TFSI 1

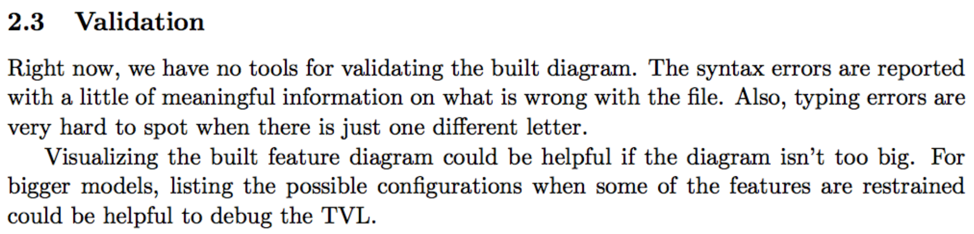
Diesel
- ○ SE 1.6 TDI 1
- ○ Sport 1.6 TDI 1
- ○ S line 1.6 TDI 1

1 Model | 2 Engine | 3 Exterior | 4 Interior

Open issues

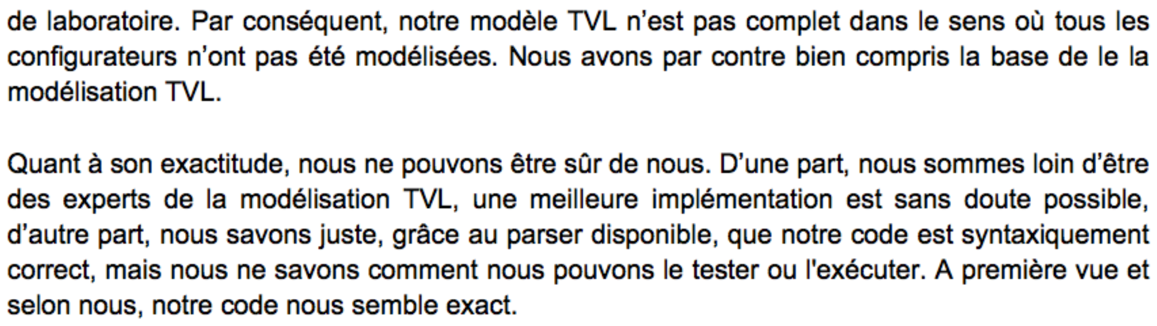
- Quality of the feature model?
  - Completeness
  - Correctness

Support for validation

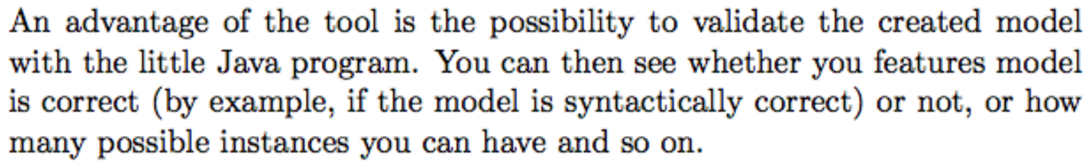

2.3 Validation

Right now, we have no tools for validating the built diagram. The syntax errors are reported with a little of meaningful information on what is wrong with the file. Also, typing errors are very hard to spot when there is just one different letter.

Visualizing the built feature diagram could be helpful if the diagram isn't too big. For bigger models, listing the possible configurations when some of the features are restrained could be helpful to debug the TVL.


de laboratoire. Par conséquent, notre modèle TVL n'est pas complet dans le sens où tous les configurateurs n'ont pas été modélisées. Nous avons par contre bien compris la base de le la modélisation TVL.

Quant à son exactitude, nous ne pouvons être sûr de nous. D'une part, nous sommes loin d'être des experts de la modélisation TVL, une meilleure implémentation est sans doute possible, d'autre part, nous savons juste, grâce au parser disponible, que notre code est syntaxiquement correct, mais nous ne savons comment nous pouvons le tester ou l'exécuter. A première vue et selon nous, notre code nous semble exact.

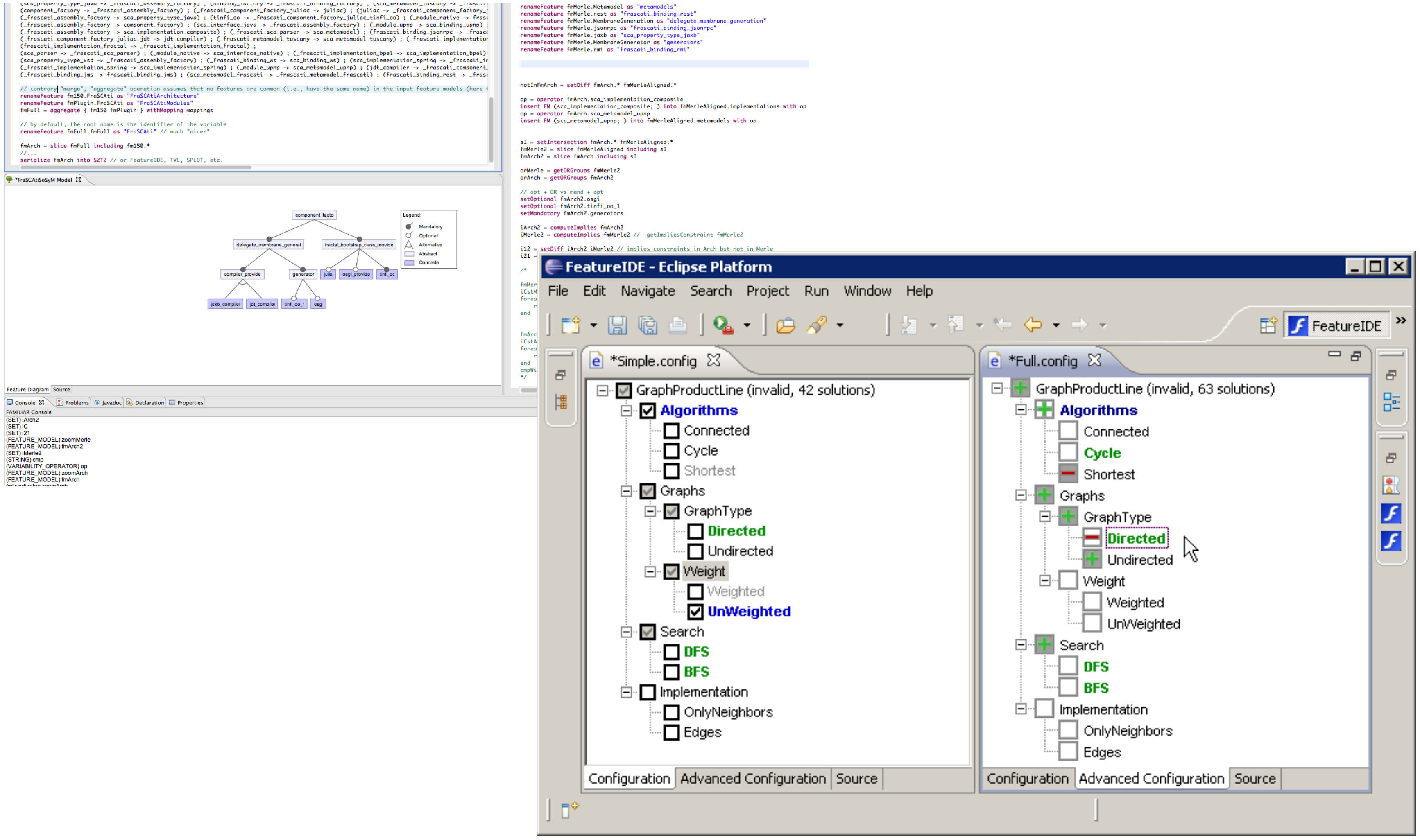

An advantage of the tool is the possibility to validate the created model with the little Java program. You can then see whether you features model is correct (by example, if the model is syntactically correct) or not, or how many possible instances you can have and so on.

Today: Feature models in depth

- **Automated reasoning**
  - Support to better understand and play with your specification
- **Feature model management**
  - Existing techniques can be considered in your work (merging)

FAMILIAR language and environment

Reasoning operations implemented in a dedicated language



```
(component_factory -> _frascati_assembly_factory) ; (_frascati_component_factory_juliac -> juliac) ; (juliac -> _frascati_component_factory_
(_frascati_assembly_factory -> sca_property_type_java) ; (tinfi_oo -> _frascati_component_factory_juliac_tinfi_oo) ; (_module_native -> frasc
(_frascati_assembly_factory -> component_factory) ; (sca_interface_java -> _frascati_assembly_factory) ; (_module_upnp -> sca_binding_upnp) ;
(_frascati_assembly_factory -> sca_implementation_composite) ; (_frascati_sca_parser -> sca_metamodel) ; (frascati_binding_jsonrpc -> _frasc
(_frascati_component_factory_juliac_jdt -> jdt_compiler) ; (_frascati_metamodel_tuscany -> sca_metamodel_tuscany) ; (_frascati_implementatio
(frascati_implementation_fractal -> _frascati_implementation_fractal) ;
(sca_parser -> _frascati_sca_parser) ; (_module_native -> sca_interface_native) ; (_frascati_implementation_bpel -> sca_implementation_bpel)
(sca_property_type_xsd -> _frascati_assembly_factory) ; (_frascati_binding_ws -> sca_binding_ws) ; (sca_implementation_spring -> _frascati_ir
(_frascati_implementation_spring -> sca_implementation_spring) ; (_module_upnp -> sca_metamodel_upnp) ; (jdt_compiler -> _frascati_component_
(_frascati_binding_jms -> frascati_binding_jms) ; (sca_metamodel_frascati -> _frascati_metamodel_frascati) ; (frascati_binding_rest -> _frasc

// contrary "merge", "aggregate" operation assumes that no features are common (i.e., have the same name) in the input feature models (here 
renameFeature fm150.FraSCAti as "FraSCAtiArchitecture"
renameFeature fmPlugin.FraSCAti as "FraSCAtiModules"
fmFull = aggregate { fm150 fmPlugin } withMapping mappings

// by default, the root name is the identifier of the variable
renameFeature fmFull.fmFull as "FraSCAti" // much "nicer"

fmArch = slice fmFull including fm150.*
//...
serialize fmArch into S2T2 // or FeatureIDE, TVL, SPLOT, etc.
```

*FraSCAtiSoSyM Model

Feature Diagram | Source

Console | Problems | Javadoc | Declaration | Properties

```
FAMILIAR Console
(SET) iArch2
(SET) iC
(SET) i21
(FEATURE_MODEL) zoomMerle
(FEATURE_MODEL) fmArch2
(SET) iMerle2
(STRING) cmp
(VARIABILITY_OPERATOR) op
(FEATURE_MODEL) zoomArch
(FEATURE_MODEL) fmArch
```

```
renameFeature fmMerle.Metamodel as "metamodels"
renameFeature fmMerle.rest as "frascati_binding_rest"
renameFeature fmMerle.MembraneGeneration as "delegate_membrane_generation"
renameFeature fmMerle.jsonrpc as "frascati_binding_jsonrpc"
renameFeature fmMerle.jaxb as "sca_property_type_jaxb"
renameFeature fmMerle.MembraneGenerator as "generators"
renameFeature fmMerle.rmi as "frascati_binding_rmi"


notInFmArch = setDiff fmArch.* fmMerleAligned.*

op = operator fmArch.sca_implementation_composite
insert FM (sca_implementation_composite; ) into fmMerleAligned.implementations with op
op = operator fmArch.sca_metamodel_upnp
insert FM (sca_metamodel_upnp; ) into fmMerleAligned.metamodels with op


sI = setIntersection fmArch.* fmMerleAligned.*
fmMerle2 = slice fmMerleAligned including sI
fmArch2 = slice fmArch including sI

orMerle = getORGroups fmMerle2
orArch = getORGroups fmArch2

// opt + OR vs mand + opt
setOptional fmArch2.osgi
setOptional fmArch2.tinfi_oo_1
setMandatory fmArch2.generators

iArch2 = computeImplies fmArch2
iMerle2 = computeImplies fmMerle2 //  getImpliesConstraint fmMerle2

i12 = setDiff iArch2 iMerle2 // implies constraints in Arch but not in Merle
i21 =

/*

fmMer
iCstM
forea
    r
end


fmArc
iCstA
forea
    r
end
cmpWi
*/
```

FeatureIDE - Eclipse Platform

*Simple.config

- GraphProductLine (invalid, 42 solutions)
  - [✓] Algorithms
    - [ ] Connected
    - [ ] Cycle
    - [ ] Shortest
  - Graphs
    - GraphType
      - [ ] Directed
      - [ ] Undirected
    - Weight
      - [ ] Weighted
      - [✓] UnWeighted
  - Search
    - [ ] DFS
    - [ ] BFS
  - [ ] Implementation
    - [ ] OnlyNeighbors
    - [ ] Edges

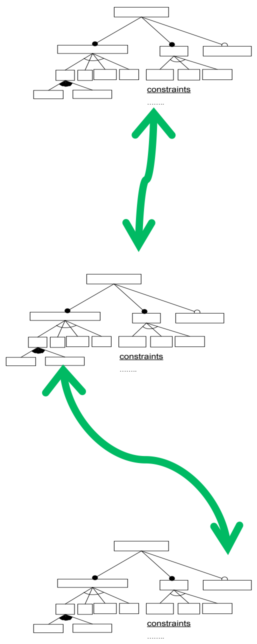
Configuration | Advanced Configuration | Source

*Full.config

- GraphProductLine (invalid, 63 solutions)
  - Algorithms
    - [ ] Connected
    - [ ] Cycle
    - Shortest
  - Graphs
    - GraphType
      - Directed
      - Undirected
    - [ ] Weight
      - [ ] Weighted
      - [ ] UnWeighted
  - Search
    - [ ] DFS
    - [ ] BFS
  - [ ] Implementation
    - [ ] OnlyNeighbors
    - [ ] Edges

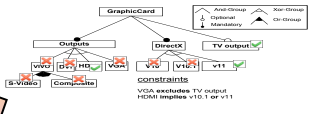
Configuration | Advanced Configuration | Source

FAMILIAR language and environment

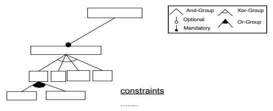


```
// foo.fml
fm1 = FM (“foo1.tvl”)
fm2 = FM (“foo2.m”)
fm3 = merge intersection { fm1 fm2 }
c3 = counting fm3
renameFeature fm3.TV as “OutputTV”
fm5 = aggregate { fm3 FM (“foo4.xml”) }
assert (isValid fm5)
fm6 = slice fm5 including fm5.TV.*
export fm6
```

FAMILIAR



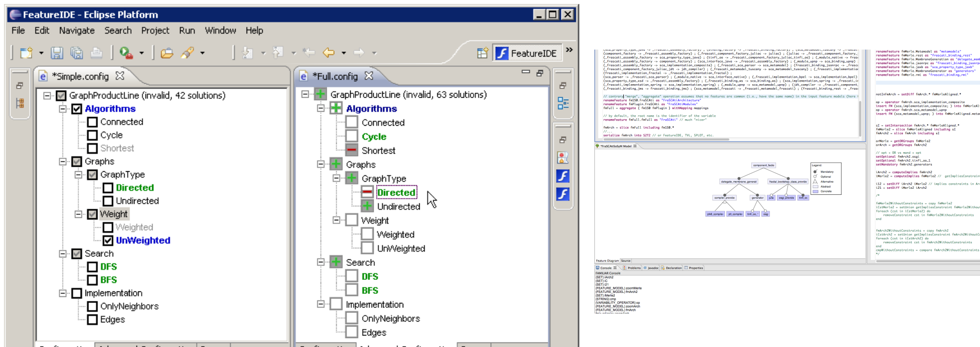
True/False
8759
“OutputTV”, “TV”



Interoperability

Language facilities

Environment


FeatureIDE - Eclipse Platform
File Edit Navigate Search Project Run Window Help
FeatureIDE
*Simple.config
*Full.config

GraphProductLine (invalid, 42 solutions)
- Algorithms
  - Connected
  - Cycle
  - Shortest
- Graphs
  - GraphType
    - Directed
    - Undirected
  - Weight
    - Weighted
    - UnWeighted
- Search
  - DFS
  - BFS
- Implementation
  - OnlyNeighbors
  - Edges

GraphProductLine (invalid, 63 solutions)
- Algorithms
  - Connected
  - Cycle
  - Shortest
- Graphs
  - GraphType
    - Directed
    - Undirected
  - Weight
    - Weighted
    - UnWeighted
- Search
  - DFS
  - BFS
- Implementation
  - OnlyNeighbors
  - Edges

- https://nyx.unice.fr/projects/familiar/

- https://nyx.unice.fr/projects/familiar/wiki/readme

No course: lab session

- Interactive of course
- 20% of the final mark
- Key idea: use, learn, criticize FAMILIAR while producing something at the end
- 3 mini-projects are proposed
  - FML Tutorial
    - Write a FAMILIAR tutorial (better than the existing one, you can be creative!), including the associated scripts
  - FML Manual
    - Modify, extend the existing one
    - Develop associated scripts and explanations
  - FML scripts repository
    - Collect all the scripts
    - Develop new scripts
      - For finding bugs + test regression script
      - Implement Benavides et al., 2010 operations (if possible)