

Next-Generation Model-based Variability Management: Languages and Tools

Mathieu Acher (Associate Professor)

Raphaël Michel (PhD candidate)

Patrick Heymans (Professor)

FUNDP NAMUR

precise

Acknowledgments

- Colleagues at University of Namur
  - Quentin Boucher, Ebrahim Abbasi, Arnaud Hubaux (PhD), Gilles Perrouin (PhD), Assoc. Prof. Anthony Cleve
- Contributors of FAMILIAR
  - Assoc. Prof. Philippe Collet, Prof. Philippe Lahire (University of Nice Sophia Antipolis)
  - Robert B. France (Colorado State University)
  - Students
    - Aleksandar Jakšić (Colorado State University)
    - Foudil Bendjabeur (University of Nice Sophia Antipolis)
    - Master students in Namur (variability management course)

Audience

- No pre-requisite background!
- Targeted Audience
  - Academics or practitioners
  - Curious guys: e.g., PhD students or modellers unaware of…
    - Variability and software product lines (SPLs)
    - Variability modelling
    - Configuration
  - MDE guys: people involved or interested in the development of model management tools
    - e.g., model composition/decomposition
  - SPL guys: advances that want to learn new technique

At the end of the tutorial…

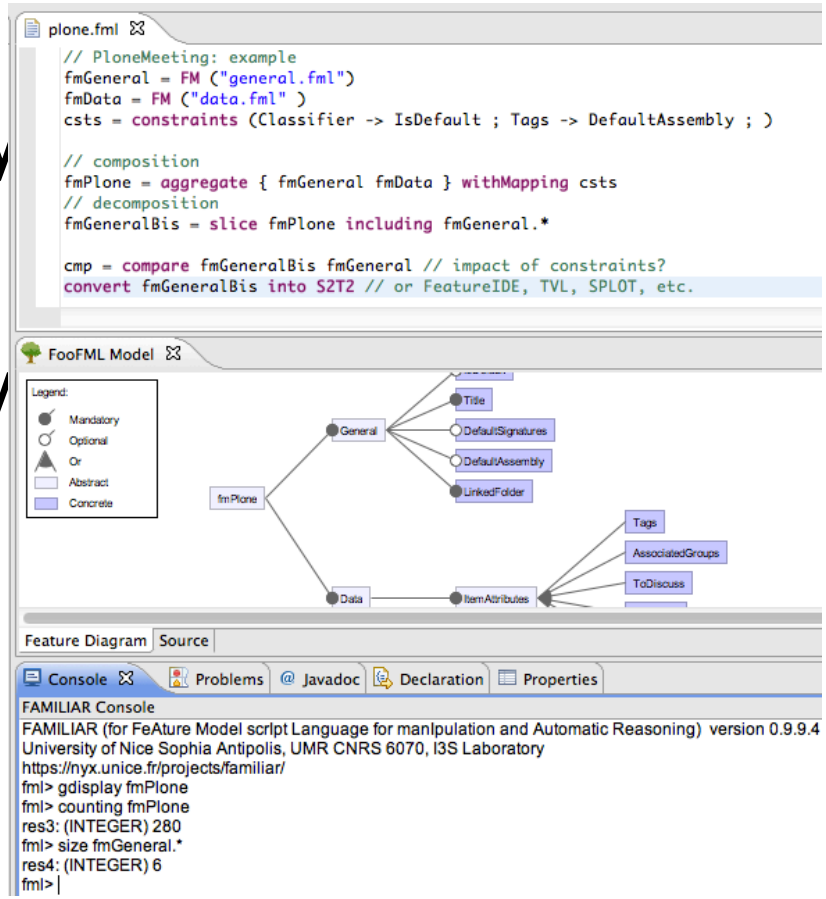
- You will have an overview of what's going on in the field of variability and model-based software product line engineering
- You will be able to go further with the languages and modelling techniques
  - reuse them in practical or academic contexts

- Supporting material: https://nyx.unice.fr/projects/familiar/wiki/SPLC12-tutorial
  - slides of the tutorial
  - related articles,
  - TVL models,
  - FAMILIAR scripts,
  - and packaged tools to interactively play with the models during the tutorial.

Plan

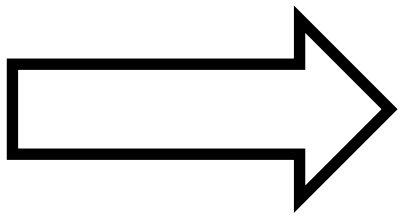
- Why managing **Variability** does matter (25’)
- **M** ity
- **M** ity (60’)

```
root PloneMeeting {
    group allof {
        General,
        Data,
        WorklowSec,
        Interface,
        Email,
        Tasks,
        Advices,
        Votes
    }
}

General {
    group allof {
        Title ,
        opt DefaultAssembly,
        opt DefaultSignatures,
        LinkedFolder,
        opt IsDefault,
        NumberLastItemLastMeeting {
            int number;
        },
        opt NumberLastMeetingConfig {
            int number;
        },
        opt MeetingConfigID
    }
}
```



0. Why managing **Variability** does (and will) matter

Software-intensive systems



come in many **variants**

Exterior | Interior

Side | Front | Rear

360°

This image may contain optional equipment. 360°

Agila, Club

1.2i 16v, 5 Speed

Blaze Red, Melt / Elba Charcoal

Total **€ 15,684.00**

1. Trims/Series | 2. Engine/Transmission | 3. Colour & Style | 4. Options | 5. Summary

Next Step

Choose Your Options

Audio/Comms/Nav | Heating/Ventilation | Mechanical | Safety/Security | **A-Z**

Audio/Comms/Nav

☑ **CD 30** - MP3 CD player with MP3 format, stereo radio, steering wheel mounted audio controls	**Standard**

Heating/Ventilation

☑ **Air conditioning**	**€ 923.00**

Mechanical

☑ **Electronic Stability Programme (ESP)**	**€ 411.00**

Safety/Security

☑ **Emergency tyre inflation kit in lieu of space-saver spare wheel and tyre**	**Standard**

Audio/Comms/Nav | Heating/Ventilation | Mechanical | Safety/Security | **A-Z**

Next Step: Summary

Legend

- ☑ Selected Option
- ☐ Selectable Option
- ☑ Option contained in an option pack
- ☐ Option contained in an option pack or standard equipment which has been replace by another option
- ▬ Option that is only selectable together with another option. Please click for details

Pricing Details

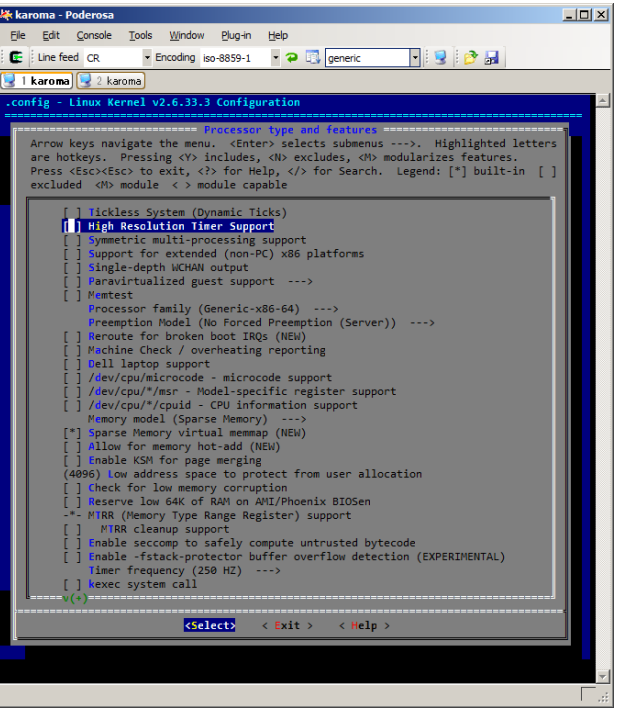
Club	€ 14,350.00
1.2i 16v, 5 Speed	
Blaze Red	€ 0.00
Melt / Elba Charcoal	€ 0.00
15-inch steel wheels with 185/60 R 15 tyres and flush wheel covers	€ 0.00

Options (2) −

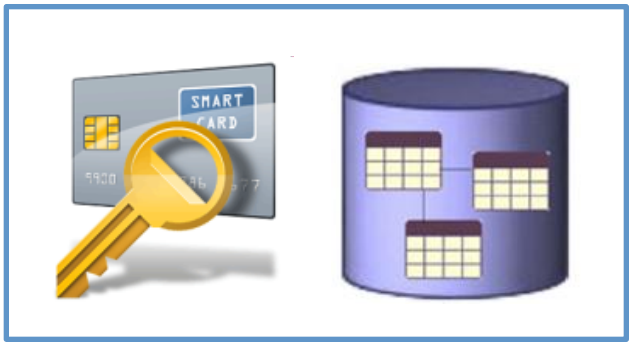
You selected:

☑ Air conditioning	€ 923.00
☑ Electronic Stability Programme (ESP)	€ 411.00
Total	**€ 15,684.00**

Linux Kernel

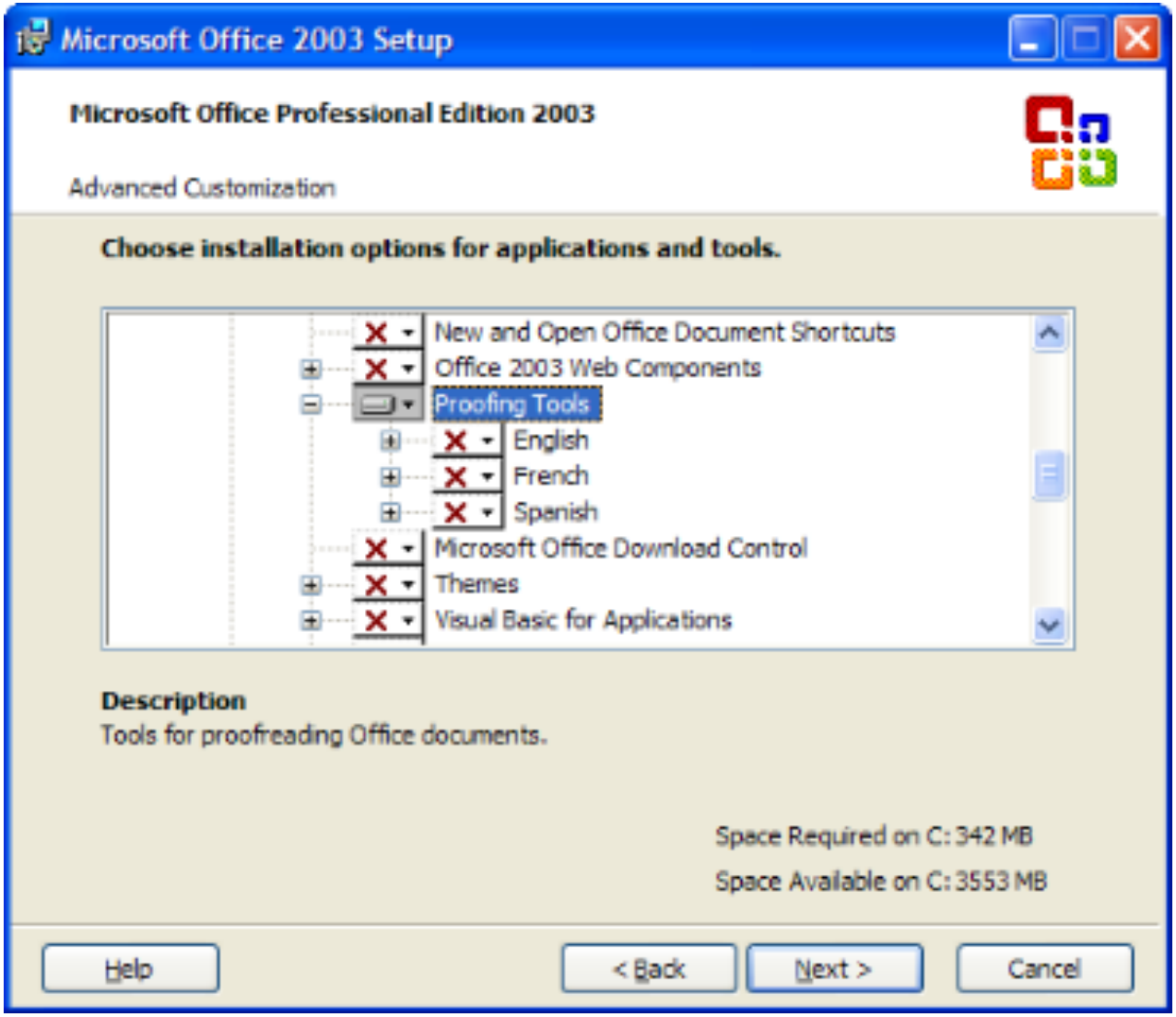


Database Engine



Printer Firmware

Features in Microsoft Office



Variability

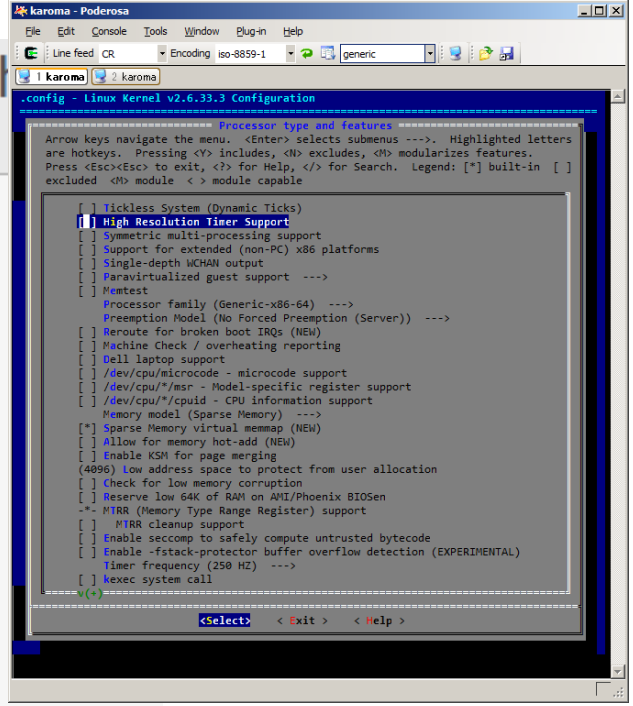
"the ability of a system to be efficiently extended, changed, customized or configured for use in a particular context"

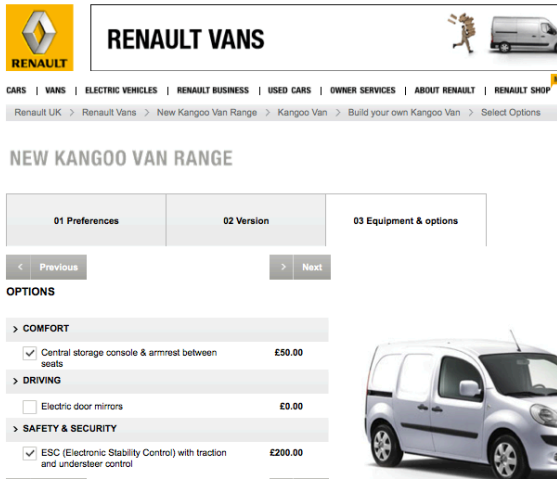
Mikael Svahnberg, Jilles van Gurp, and Jan Bosch (2005)

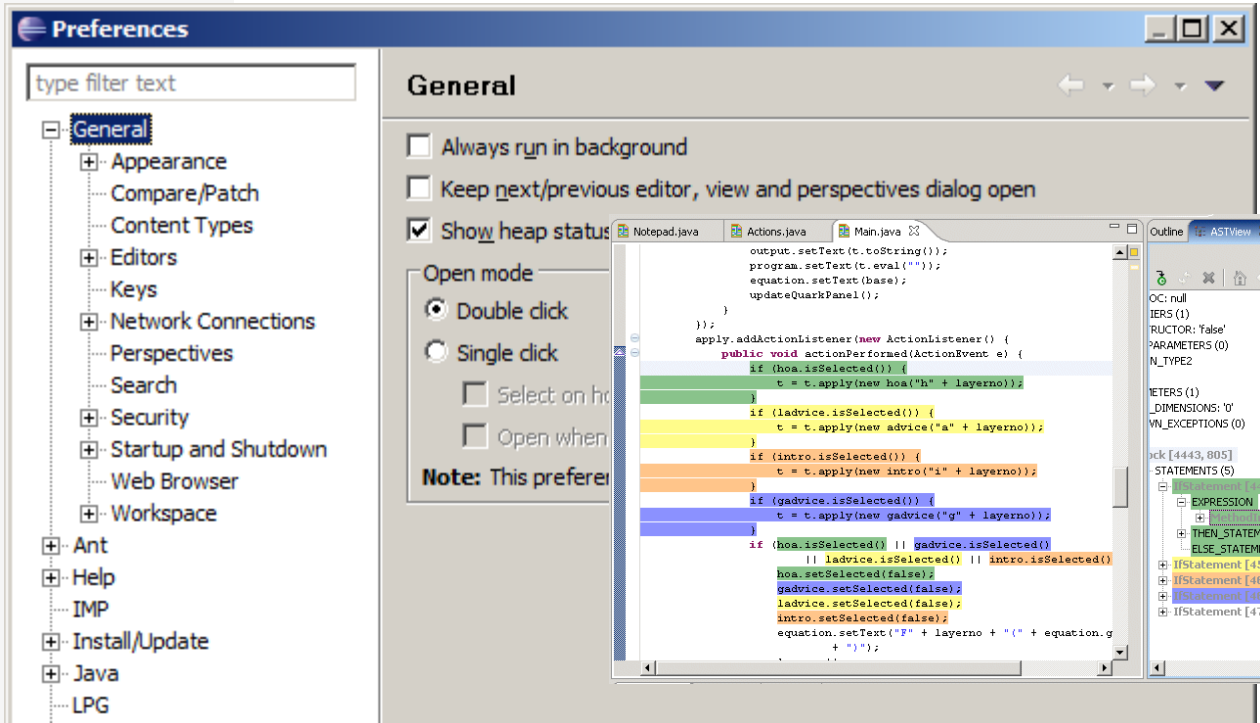
httpd.conf -- win32 Apach

Building a Web Server, for Windows

```
Listen 80
ServerRoot "/www/Apache2"
DocumentRoot "/www/webroot"

ServerName localhost:80
ServerAdmin admin@localhost

ServerSignature On
ServerTokens Full

DefaultType text/plain
AddDefaultCharset ISO-8859-1

UseCanonicalName Off

HostnameLookups Off

ErrorLog logs/error.log
LogLevel error

PidFile logs/httpd.pid

Timeout 300

KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 15


<IfModule mpm_winnt.c>
  ThreadsPerChild 250
  MaxRequestsPerChild 0
</IfModule>
```







If you're able to master variability…

- Reduce development costs
- Reduce certification costs
- Shorten time-to-market

- But, are you able?
  - developing, verifying, certifying billions of variants is challenging!

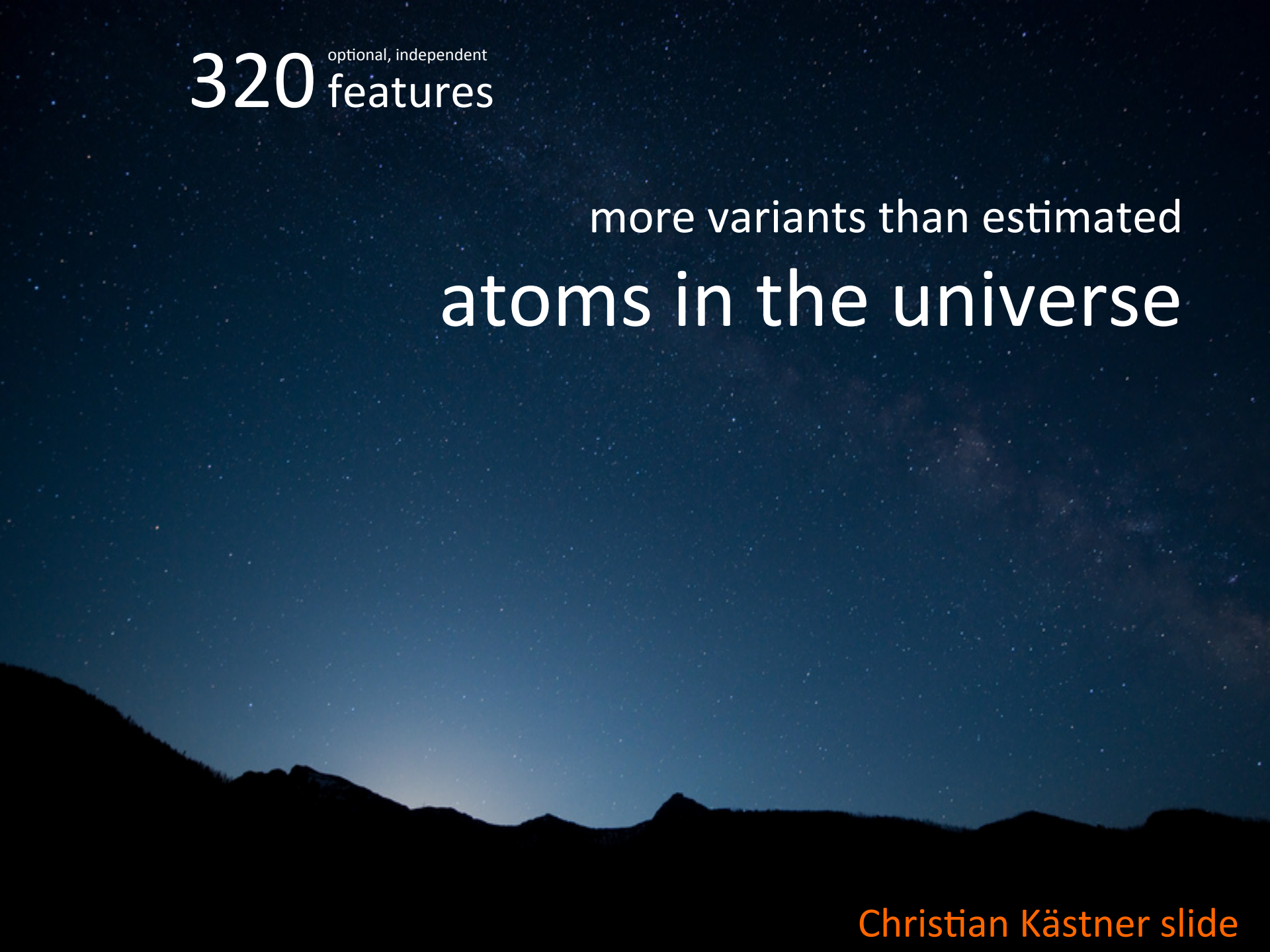

Variability = Complexity
Christian Kästner slide

33 optional, independent features

a unique variant for every

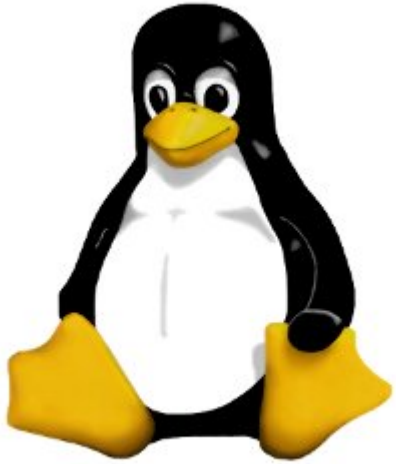
person on this planet

Christian Kästner slide


320 optional, independent features
more variants than estimated
atoms in the universe
Christian Kästner slide

2000 features

10000 features



Christian Kästner slide


Automation?
Avoid solving the same problem!
2, 3…n times

Christian Kästner slide

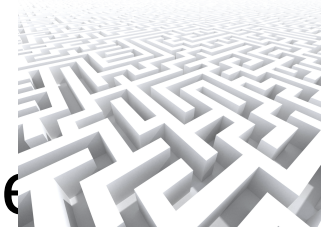
Maintenance?
Comprehension?

Why managing Variability does (and will) matter

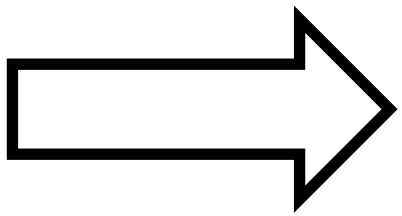
Goal: Software mass customization / Adaptive and configurable systems

Problem: Variability = Complexity

Approach: Model-based variability manageme

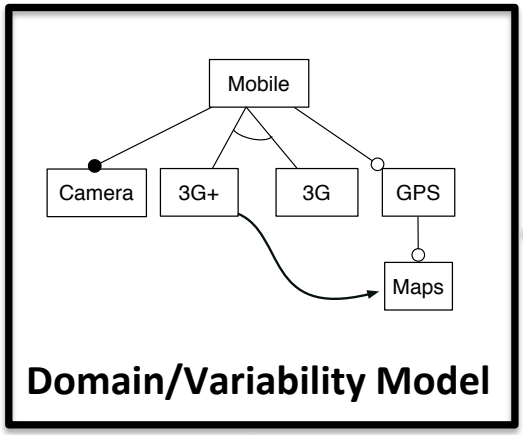


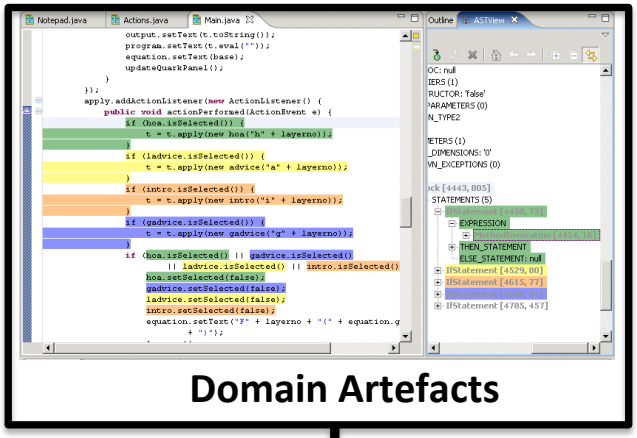
Software-intensive systems come in many variants



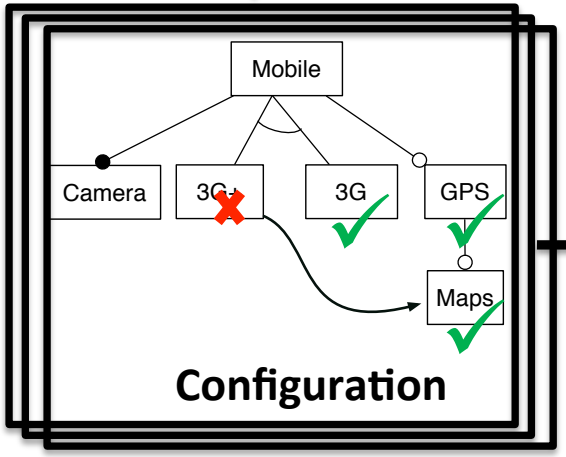
Model-based Variability Management

Domain Engineering



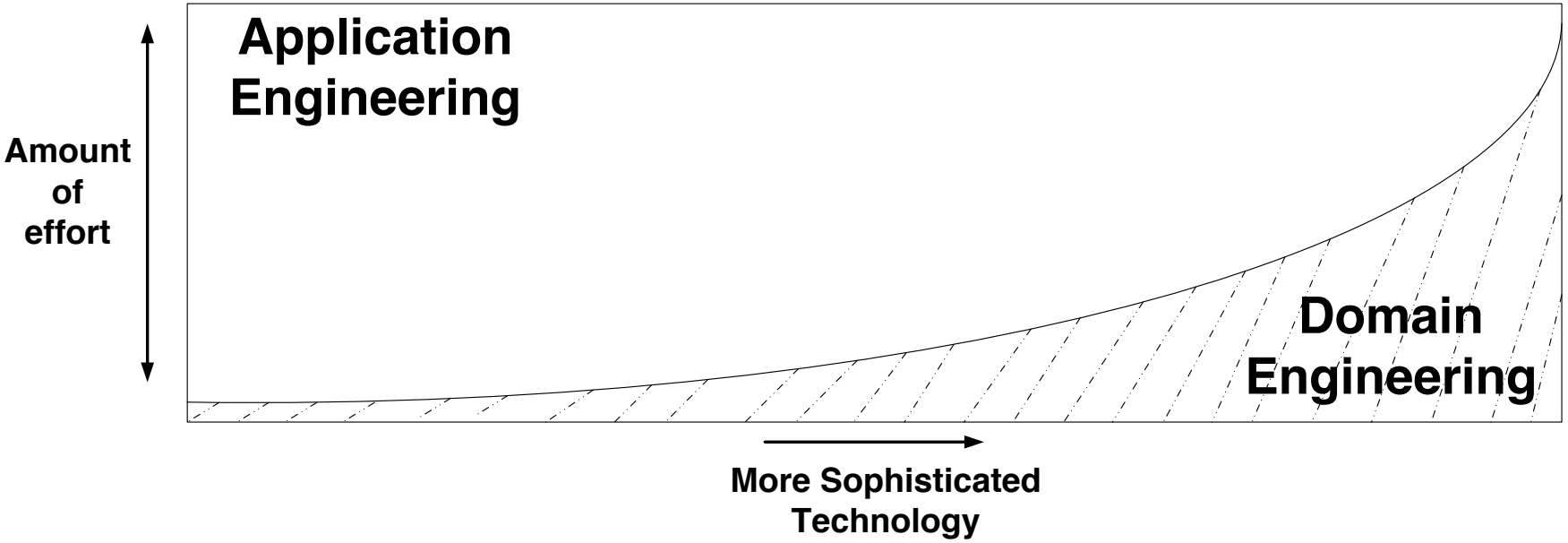


Application Engineering



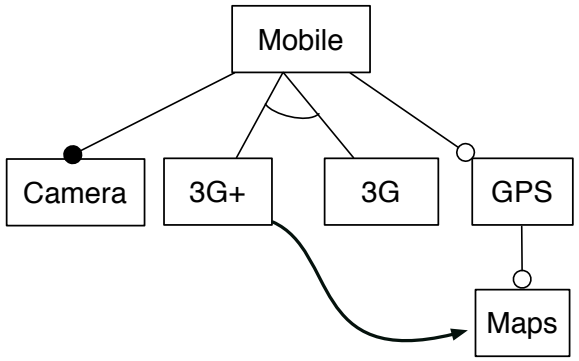
*« the investments required to develop the reusable artifacts during **domain engineering**, are outweighed by the benefits of deriving the individual products during **application engineering** »*

Jan Bosch et al. (2004)

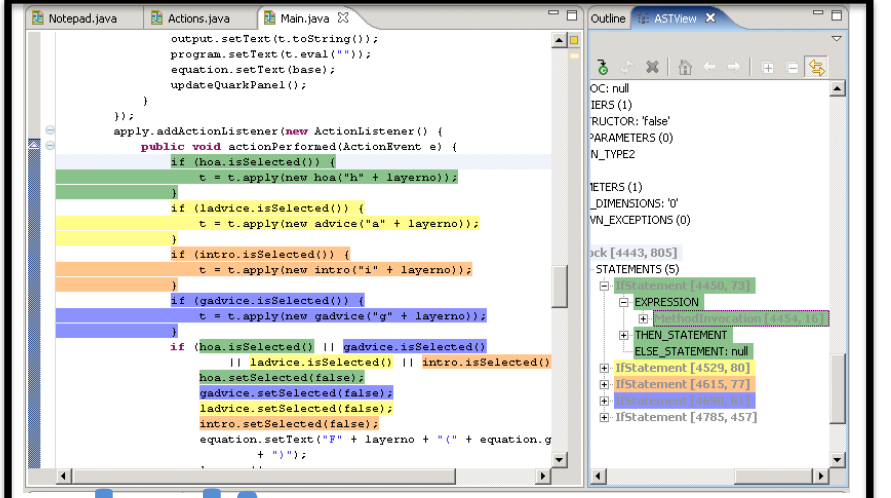


99% domain engineering, 1% application engineering?

- specifies what you want (click, click, click) a customized product is automatically built for you
- Iterate the process for *n* products

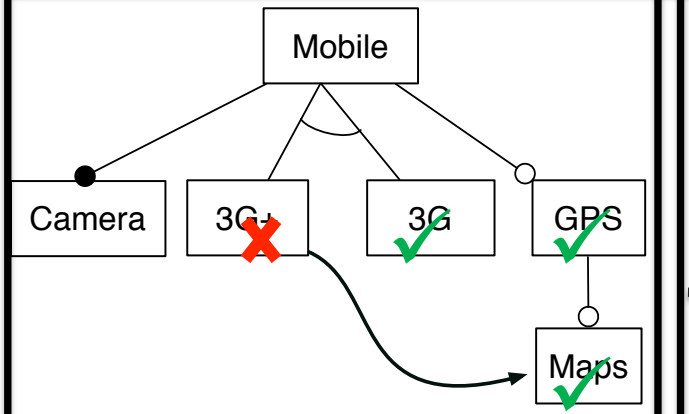


Variability Model



Domain Artefacts (e.g., source code)

Modeling variability is crucial



Configuration

Software Generator


Unused flexibility

Illegal variant

Variability Model

Feature Model: defacto standard

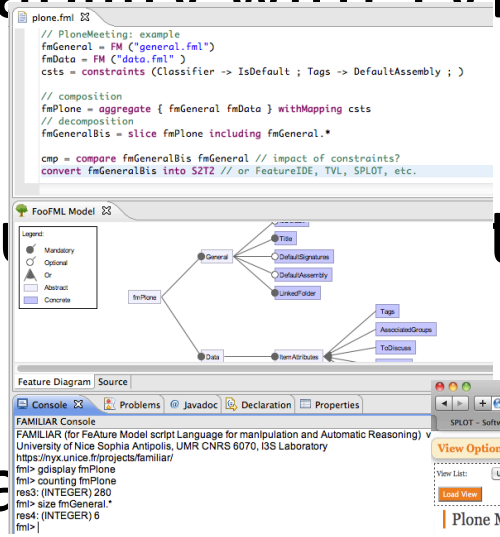
- Research
  - 2500+ citations of [Kang et al., 1990] on Google Scholar
  - Central to many generative approaches
    - at requirements or code level
  - Tools & Languages (GUIDSL/FeatureIDE, SPLOT, FaMa, etc.)
- Industry
  - Tools (Gears, pure::variants),
  - Will be Part of Common Variability Language (CVL), future OMG standard

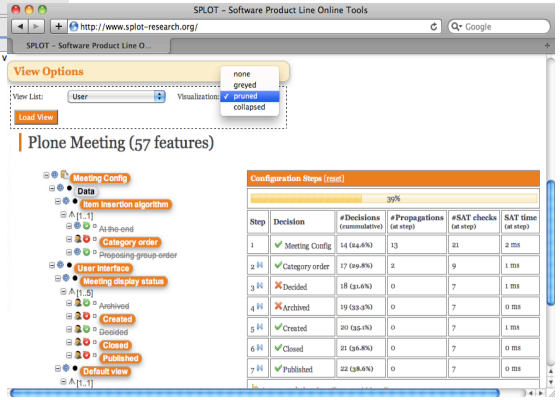
Plan

- W… g **Variability** does matter
- **M**…jability with TVL
- **M**…th FAMILIAR
- **Configuring** fea…

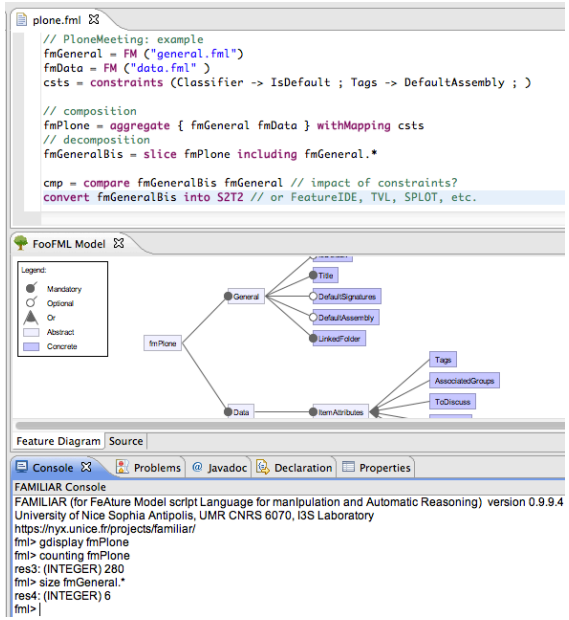
```
root PloneMeeting {
    group allof {
        General,
        Data,
        WorklowSec,
        Interface,
        Email,
        Tasks,
        Advices,
        Votes
    }
}
```

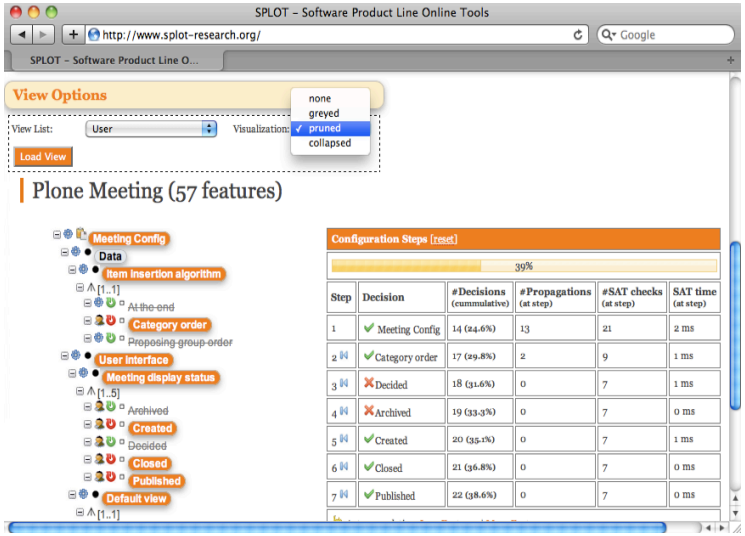
```
General {
    group allof {
        Title ,
        opt DefaultAssembly,
        opt DefaultSignatures,
        LinkedFolder,
        opt IsDefault,
        NumberLastItemLastMeeting {
            int number;
        },
        opt NumberLastMeetingConfig {
            int number;
        },
        opt MeetingConfigID
    }
}
```





```
root PloneMeeting {
    group allof {
        General,
        Data,
        WorklowSec,
        Interface,
        Email,
        Tasks,
        Advices,
        Votes
    }
}

General {
    group allof {
        Title ,
        opt DefaultAssembly,
        opt DefaultSignatures,
        LinkedFolder,
        opt IsDefault,
        NumberLastItemLastMeeting {
            int number;
        },
        opt NumberLastMeetingConfig {
            int number;
        },
        opt MeetingConfigID
    }
}
```





Formally define the scope of your product line with a textual language

Edit, compose, decompose, compare, reason about your variability specifications

Configure your product line with a collaborative, distributed and dependable feature-based configuration

1. Modelling variability with TVL

(Text-based Variability Language)

Audi
Vorsprung durch Technik

R8 Spyder

5.2 FSI quattro R tronic

Prix total

171.216,00 EUR

Prix de base	**170.490,00 EUR**
Equipements optionnels	**726,00 EUR**

- Informations détaillées
- Entrez l'Audi Code
- Générer un PDF
- Nouvelle configuration

[+] Plein écran / Dimensions | Fermer la capote | Habitacle | Tableau de bord

Packs

Aucun pack n'est proposé pour ce modèle.

Couleurs

Blanc Ibis
Noir
Prix: 0,00 EUR

Couleurs métallisées à partir de 0,00 EUR

Couleurs à effet perlé à partir de 0,00 EUR

Couleurs personnalisées Audi exclusive

Audi exclusive

Couleur capote

Noir

Jantes

4 Jantes alu 5 BRANCHES ROTOR finition titane 8,5 x 19 à l'avant, 11 x 19 à l'arrière. Pneus 235/35 R19 à l'avant et 305 /30 R19 à l'arrière
Prix: 726,00 EUR

19" à partir de 0,00 EUR

1 Modèle | 2 Moteur | 3 Extérieur | 4 Intérieur | 5 Option | 6 Votre Audi

Audi
Vorsprung durch Technik

R8 Spyder
5.2 FSI quattro R tronic

Prix total

185.899,35 EUR

Prix de base	**170.490,00 EUR**
Equipements optionnels	**15.409,35 EUR**

- Informations détaillées
- Entrez l'Audi Code
- Générer un PDF
- Nouvelle configuration

[+] Plein écran / Dimensions Vue extérieure Tableau de bord

- Packs d'équipements
- Extérieur
- Jantes & pneumatiques
- Intérieur
- Volants
- Sièges
- **Sécurité & technique**
- Infotainment

- Châssis
- Freins
- **Systèmes d'assistance**
- Autres

excludes

☑ Régulateur de vitesse		320,65 EUR
☐ Système d'aide au stationnement APS avant / arrière		931,70 EUR
☐ Système d'aide au stationnement APS avant / arrière avec affichage dans l'écran MMI		1.373,35 EUR
☑ Système d'aide au stationnement Advanced : APS avant et arrière et caméra arrière		1.790,80 EUR
● Audi hill assist : assistance au démarrage en côte		Série

Attention:
Le prix peut varier en fonction du choix de moteur et des équipements.

Un aperç des équipements:

Mode expert

Réinitialiser la sélection

1 Modèle 2 Moteur 3 Extérieur 4 Intérieur 5 Option 6 Votre Audi

Audi
Vorsprung durch Technik

A5 Sportback
3.0 TDI quattro S tronic

Prix total
54.460,15 EUR

Prix de base	50.570,00 EUR
Equipements optionnels	3.890,15 EUR

- Informations détaillées
- Entrez l'Audi Code
- Nouvelle configuration

[+] Plein écran / Dimensions

Vérification de votre sélection

Cet équipement nécessite un équipement complémentaire:

☑ GPS Plus avec disque dur		2.934,25 EUR

Voici les équipements complémentaires possibles:

○ Ordinateur de bord en couleur avec programme efficiency **Remarque:** uniquement sur les modèles avec système Start-Stop et uniquement disponible en combinaison avec l'autoradio Concert, l'autoradio Symphony ou un système de navigation		181,50 EUR
○ Pack Intenso Plus		3.100,00 EUR

- Extérieur
- Jantes & pneumatiques
- Intérieur
- Volants
- Sièges
- Sécurité & technique
- **Infotainment**

2.934,25 EUR

● Sans appareil de navigation — Série

Attention:
Le prix peut varier en fonction du choix de moteur et des équipements.

Un aperç des équipements:

Mode expert

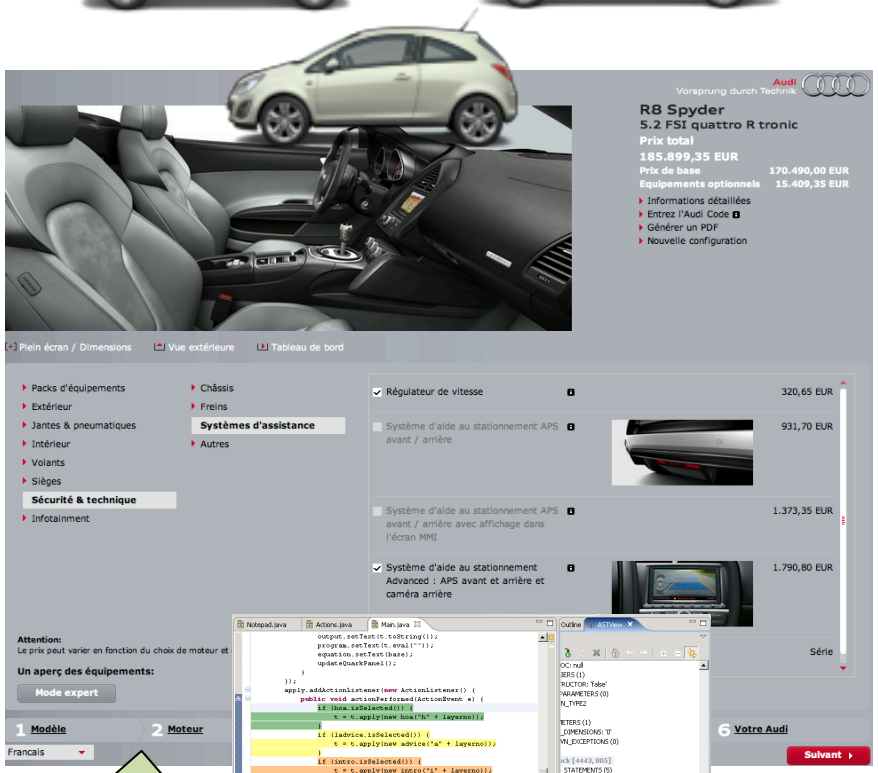
Réinitialiser la sélection

1 Modèle | 2 Moteur | 3 Extérieur | 4 Intérieur | 5 Option | 6 Votre Audi

Francais

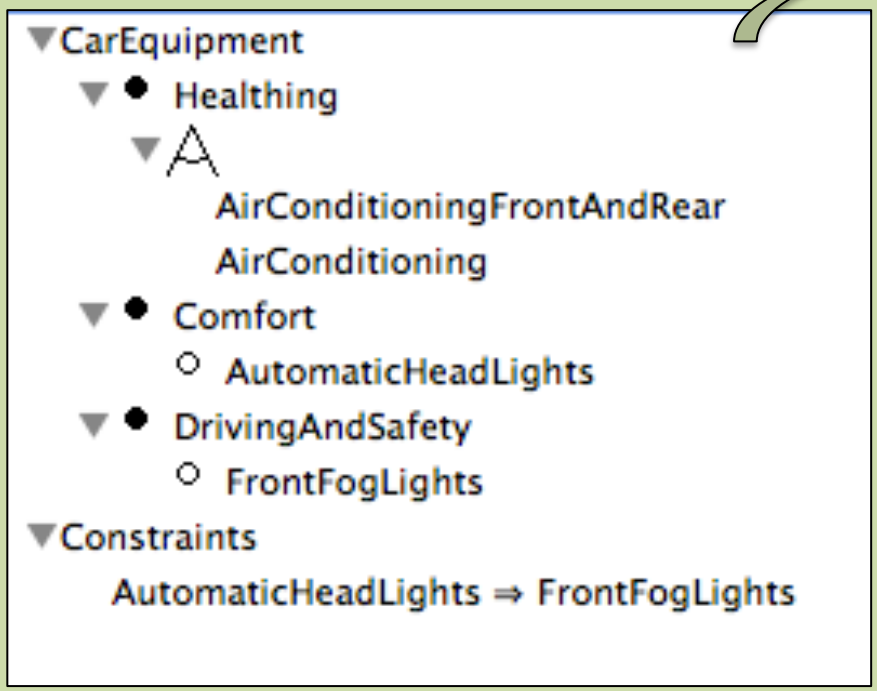
Suivant

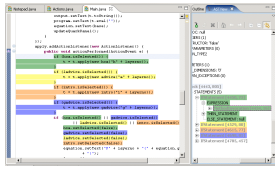
Feature Models



- CarEquipment
  - Healthing
    - AirConditioningFrontAndRear
    - AirConditioning
  - Comfort
    - AutomaticHeadLights
  - DrivingAndSafety
    - FrontFogLights
- Constraints
  - AutomaticHeadLights ⇒ FrontFogLights

Feature Models (Background)

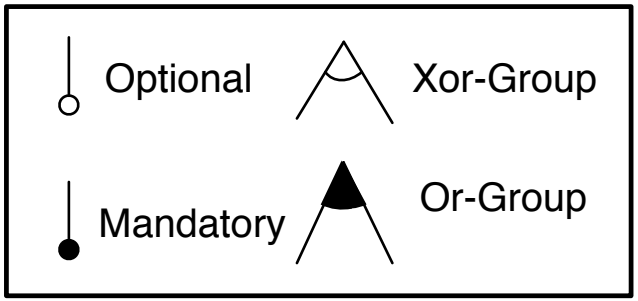


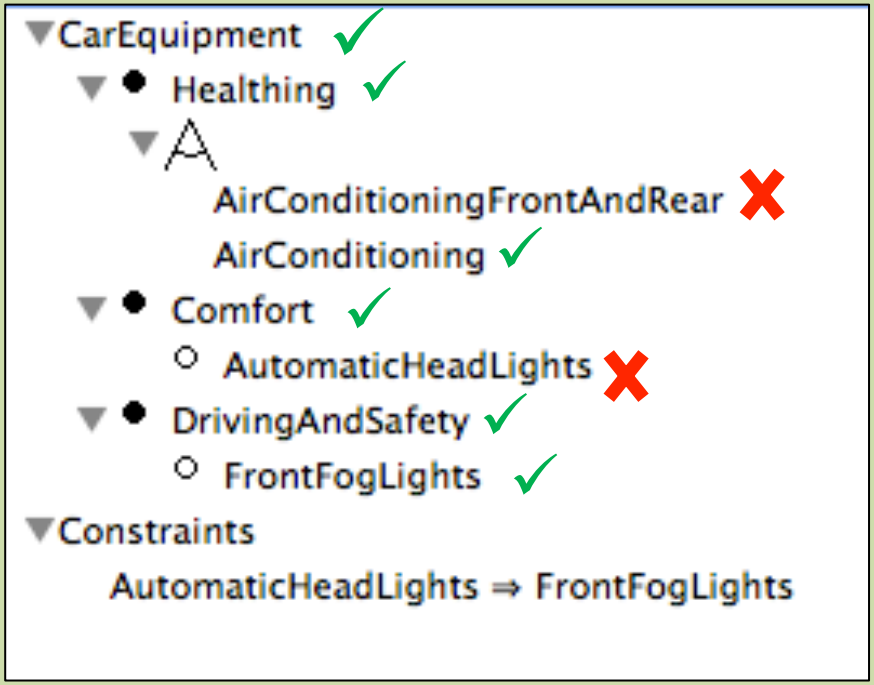


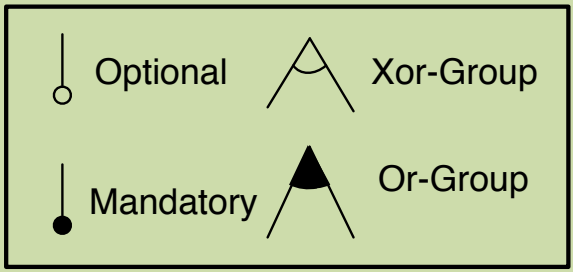
Hierarchy: rooted tree

Variability:

- mandatory,
- optional,
- Groups: exclusive or inclusive features
- Cross-tree constraints



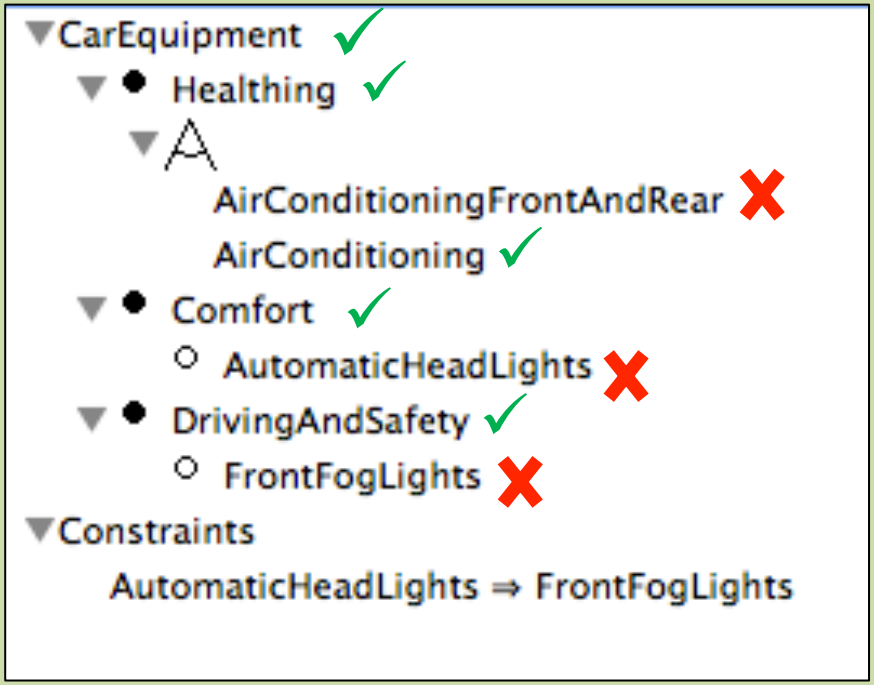


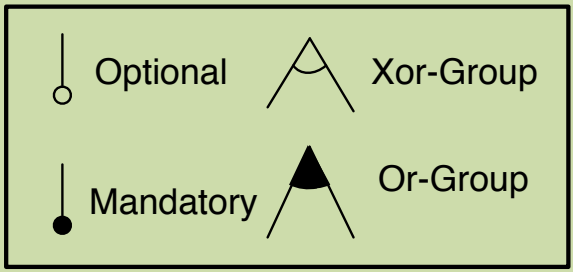


Hierarchy + Variability
=
set of valid configurations

configuration = set of features selected

{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning, FrontFogLights}

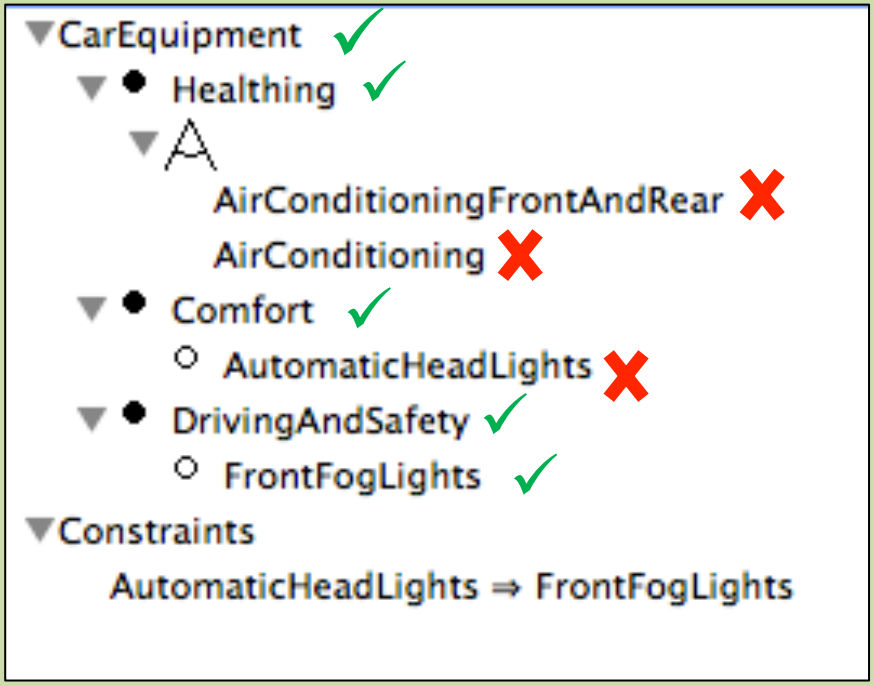


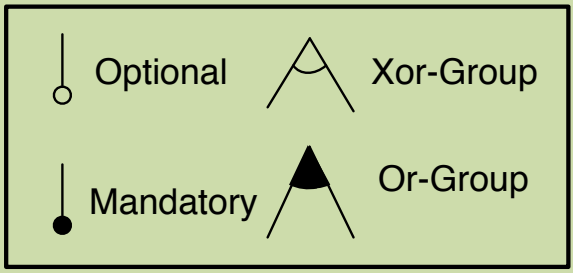


Hierarchy + Variability
=
set of valid configurations

configuration = set of features selected

{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning}

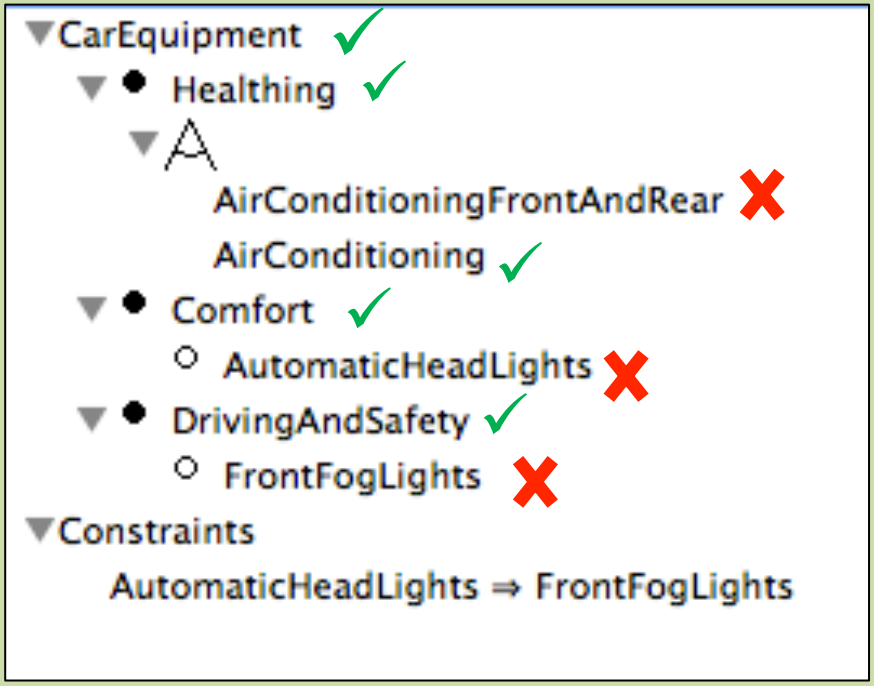


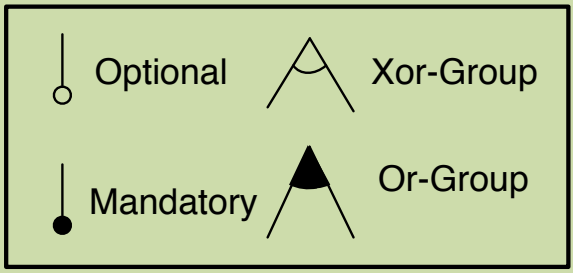


Hierarchy + Variability
=
set of valid configurations

configuration = set of features selected

{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning, AirConditioningFrontAndRear, FrontFogLights}

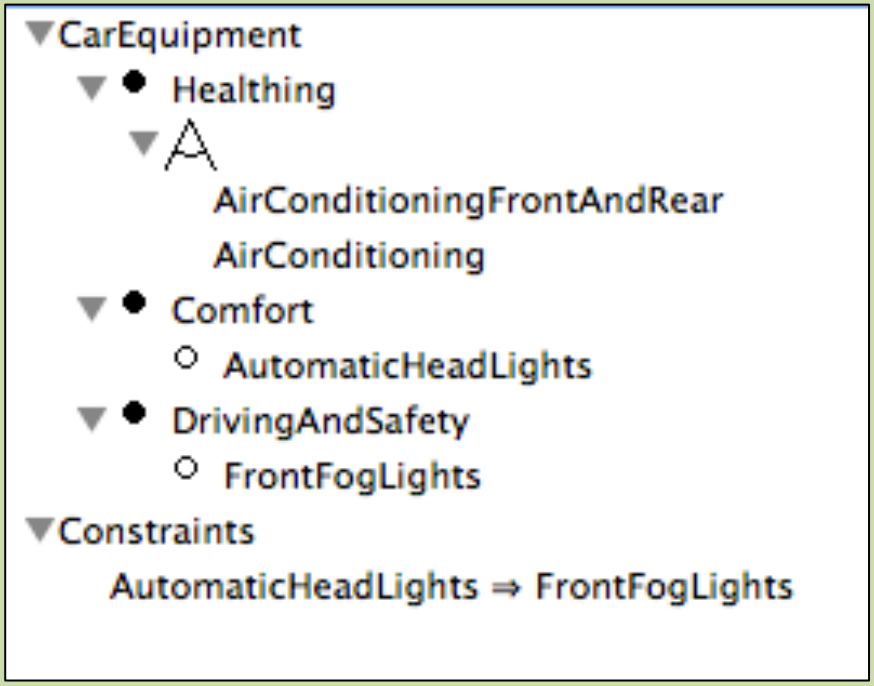


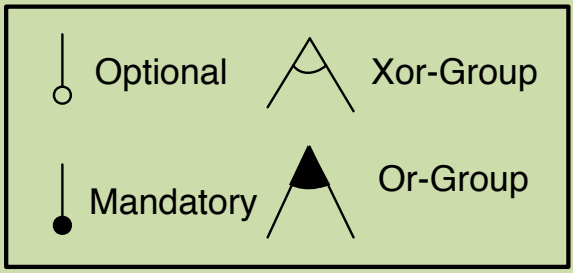


Hierarchy + Variability
=
set of valid configurations

configuration = set of features selected

{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning}





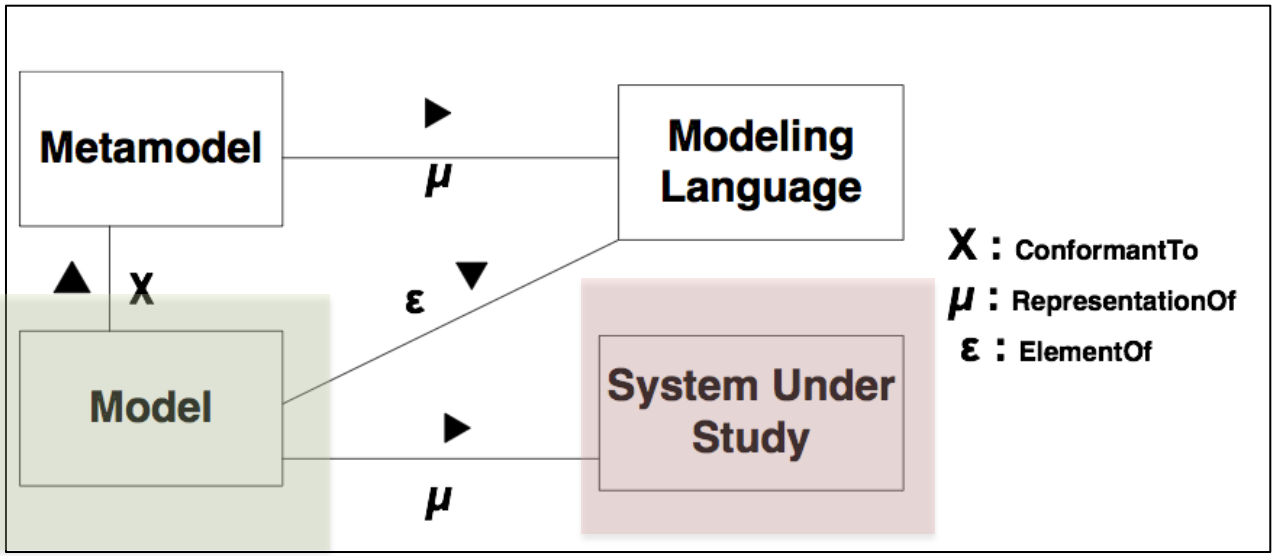
Hierarchy + Variability
=
set of valid configurations

{CarEquipment, Comfort, DrivingAndSafety, Healthing}

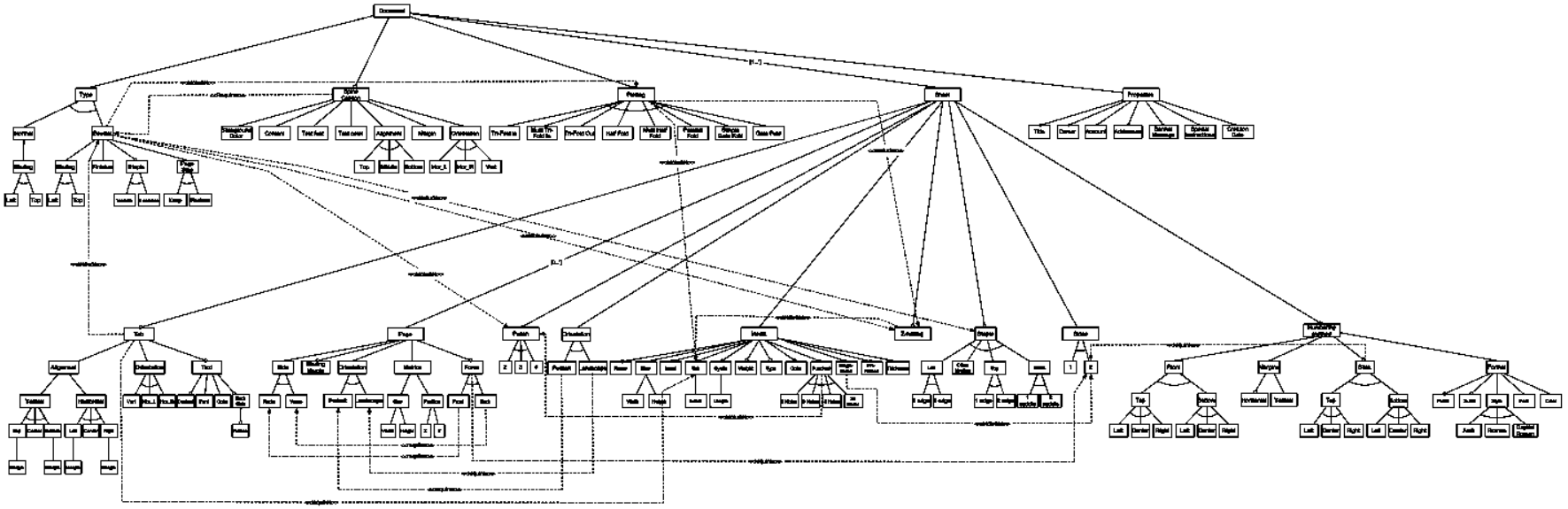
X

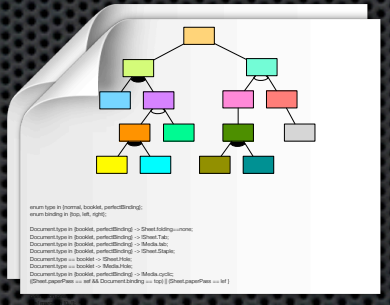
{AirConditioning, FrontFogLights}
{AutomaticHeadLights, AirConditioning, FrontFogLights}
{AutomaticHeadLights, FrontFogLights, AirConditioningFrontAndRear}
{AirConditioningFrontAndRear}
{AirConditioning}
{AirConditioningFrontAndRear, FrontFogLights}

Feature Models

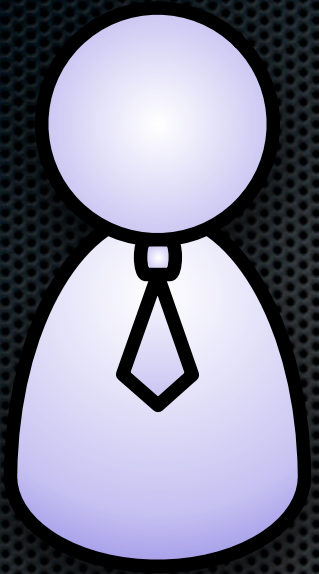


- CarEquipment
  - Healthing
    - AirConditioningFrontAndRear
    - AirConditioning
  - Comfort
    - AutomaticHeadLights
  - DrivingAndSafety
    - FrontFogLights
- Constraints
  - AutomaticHeadLights ⇒ FrontFogLights


Document
Type
Space allocation
Printing
Sheet
Proportion
1..*
Normal
Standard
Portrait
Landscape
Wrapping
Binding
Position
Staple
Plain Paper
Left
Top
Left
Top
Vertical
Horizontal
Keep
Shadow
Transparency Color
Contours
Text Fold
Text color
Alignment
Margin
Orientation
Top
Middle
Bottom
Mov_X
Mov_R
Vert
Tri-Fold In
Multi-Tri-Fold In
Tri-Fold Out
Half Fold
Multi Half Fold
Parallel Fold
Roman Gate Fold
Gate Fold
Title
Name
Number
Addressee
Special Message
Special Instructions
Creation Date
Tab
Page
Finish
Orientation
Width
Embedding
Staple
Sides
Alignment
Horizontal
Vertical
Text
Style
Multi-Up Mode
Orientation
Mirror
Rules
Position
Portrait
Landscape
Reverse
Mirror
Inset
Tile
Symbol
Weight
Type
Color Mode
Size
Name
Front
Back
Inside
Outside
Both Sides
None
Left
Center
Right
Top
Middle
Bottom
Portrait
Landscape
Reverse Portrait
Reverse Landscape
Auto
Manual
Offset
Saddle
Corner



```
enum type in {normal, booklet, perfectBinding};
enum binding in {top, left, right};

Document.type in {booklet, perfectBinding} -> Sheet.folding == none;
Document.type in {booklet, perfectBinding} -> !Sheet.Tab;
Document.type in {booklet, perfectBinding} -> !Media.tab;
Document.type in {booklet, perfectBinding} -> !Sheet.Staple;
Document.type == booklet -> !Sheet.Hole;
Document.type == booklet -> !Media.Hole;
Document.type in {booklet, perfectBinding} -> !Media.cyclic;
{|Sheet.paperPass == ref && Document.binding == top|} {|Sheet.paperPass == ref|}
```



Visualisation will help, right ?

- Tools create their own problems
- Lock-in, dependance
- Some information is inevitably textual
  - OCL constraints in UML
  - Minispecs in StateCharts
  - Attributes and constraints in FD

Why not use text altogether ?

- There are powerful tools for that
- Helps with
  - Editing
  - Transformation
  - Versioning
  - Information exchange
  - …
- Graphical views can be generated anyway

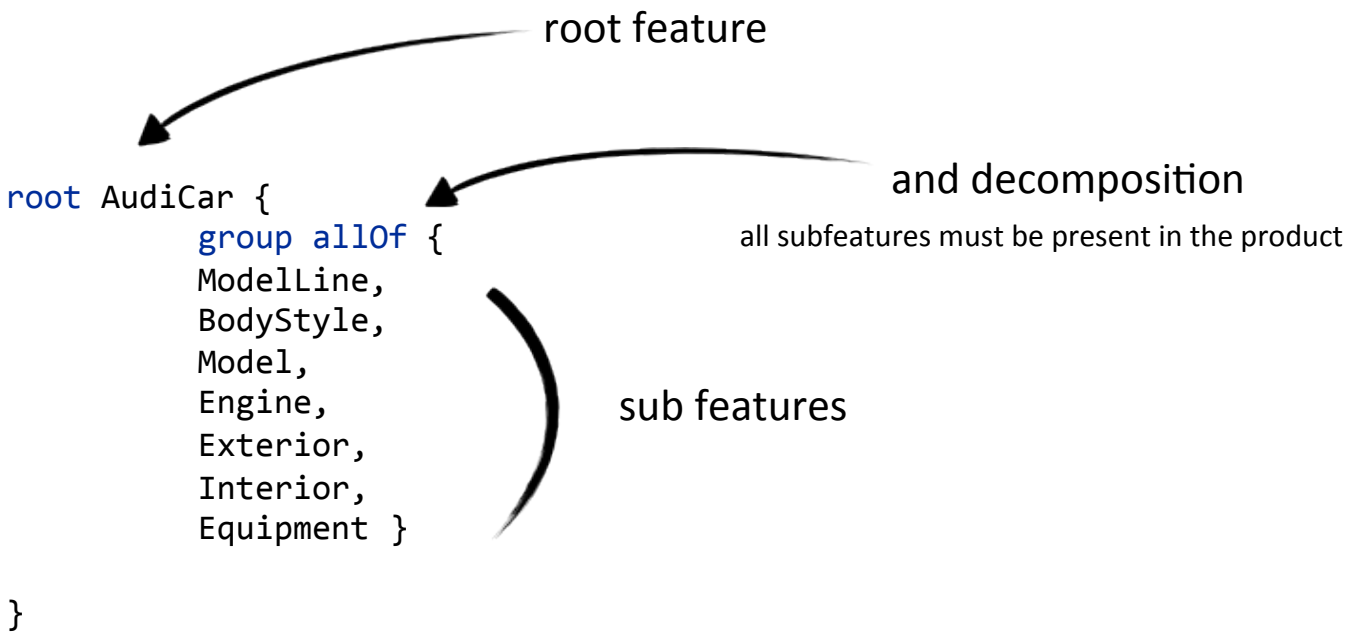
Not a new idea

Language	User Friendly	Attributes	Cardinalities	Cross tree constraints	Modula-risation
Van Deursen et al.	✔				
GUIDSL (AHEAD)	✔				
SXFM	✔				
XML-based (FAMA etc.)		✔	✔	✔	

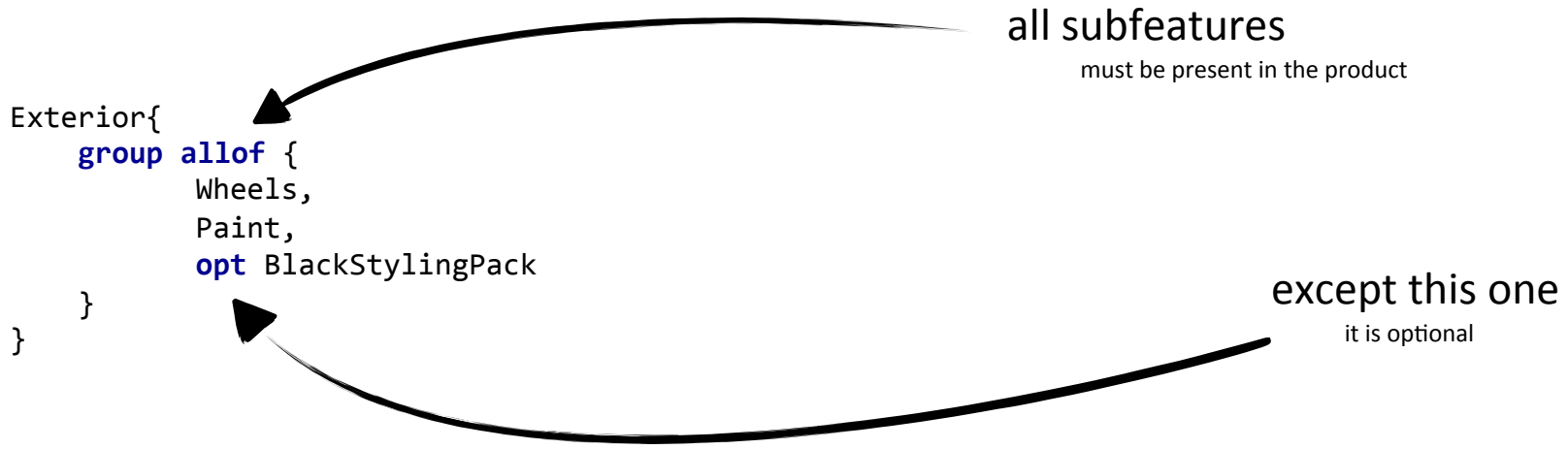
Furthermore

- Intuitive C-like syntax
- Formal semantics
- Statically typed
- Implemented

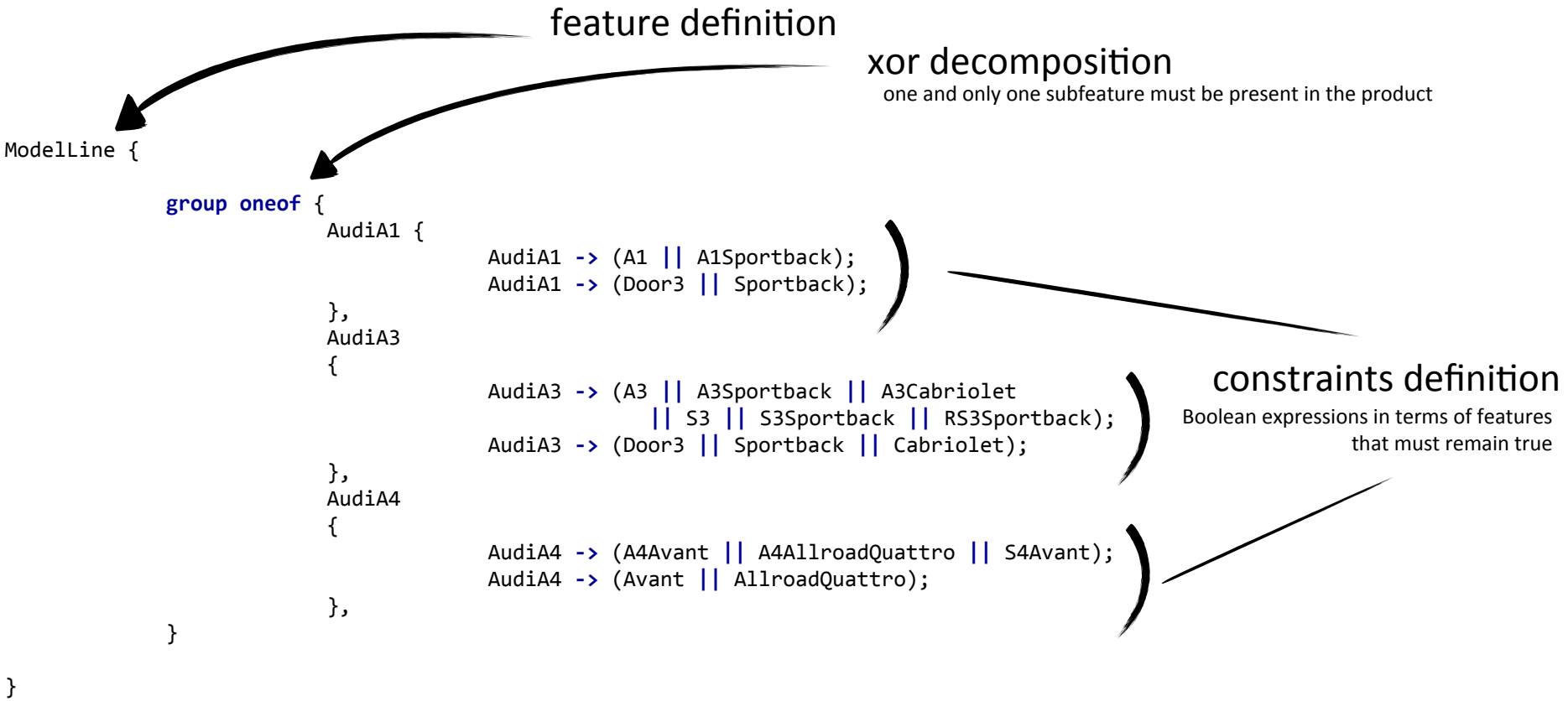
TVL : allOf – and decomposition



TVL : optional features

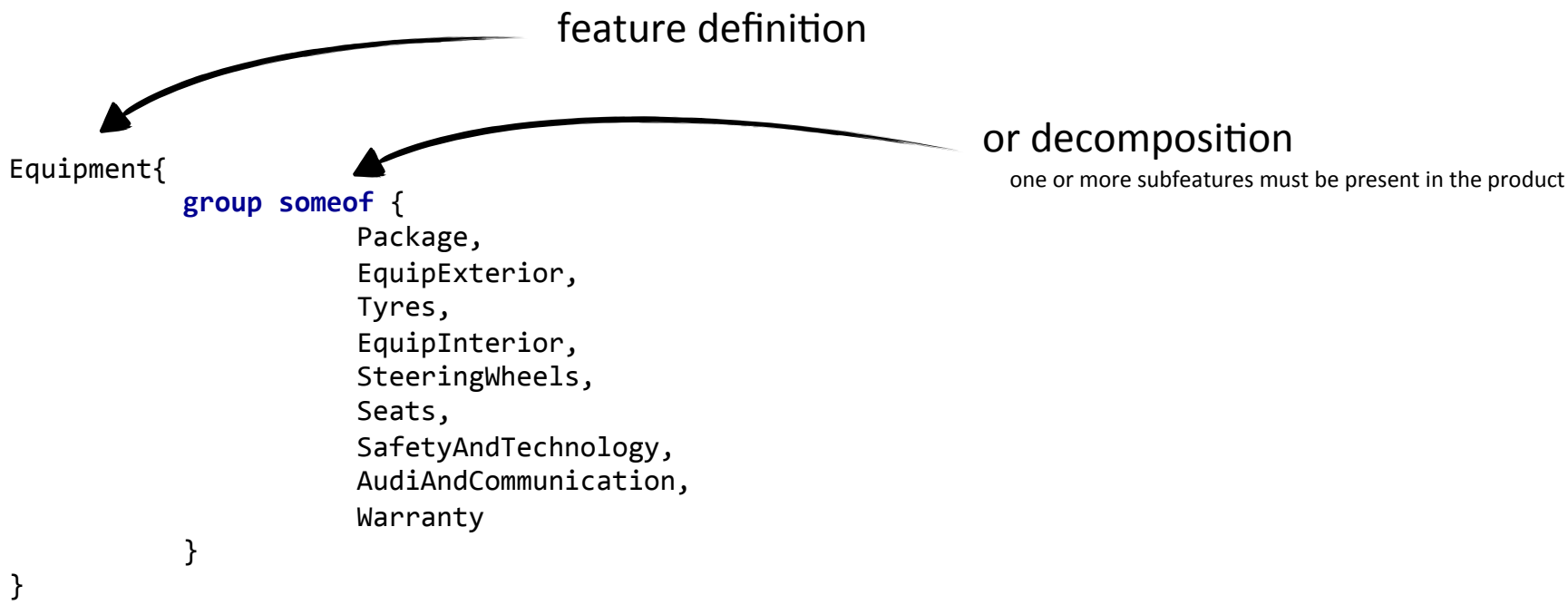


TVL : one of – xor decomposition

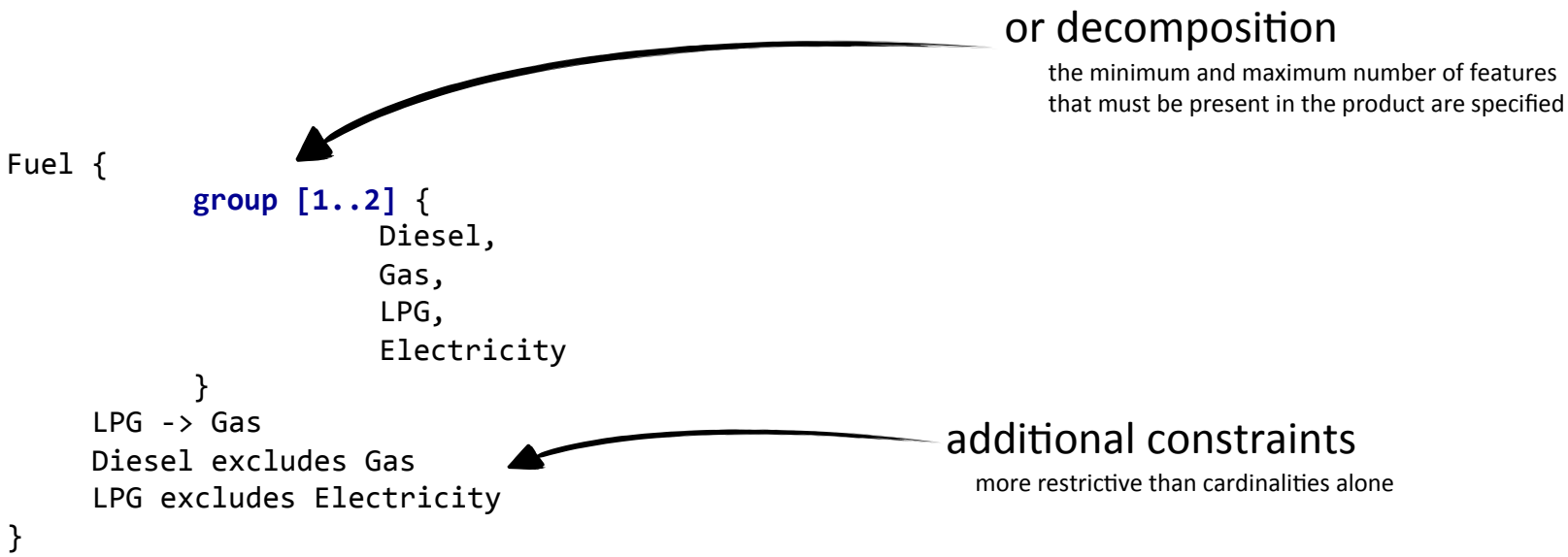



```
ModelLine {
        group oneof {
                AudiA1 {
                        AudiA1 -> (A1 || A1Sportback);
                        AudiA1 -> (Door3 || Sportback);
                },
                AudiA3
                {
                        AudiA3 -> (A3 || A3Sportback || A3Cabriolet
                                || S3 || S3Sportback || RS3Sportback);
                        AudiA3 -> (Door3 || Sportback || Cabriolet);
                },
                AudiA4
                {
                        AudiA4 -> (A4Avant || A4AllroadQuattro || S4Avant);
                        AudiA4 -> (Avant || AllroadQuattro);
                },
        }

}
```

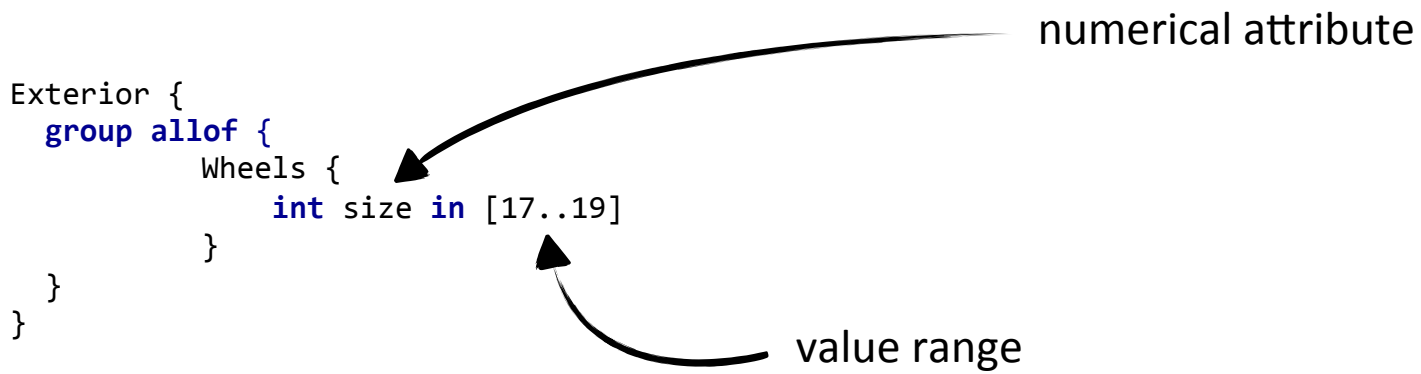
TVL : someOf – or decomposition



TVL : Cardinalities



TVL : Attributes



TVL : Constants

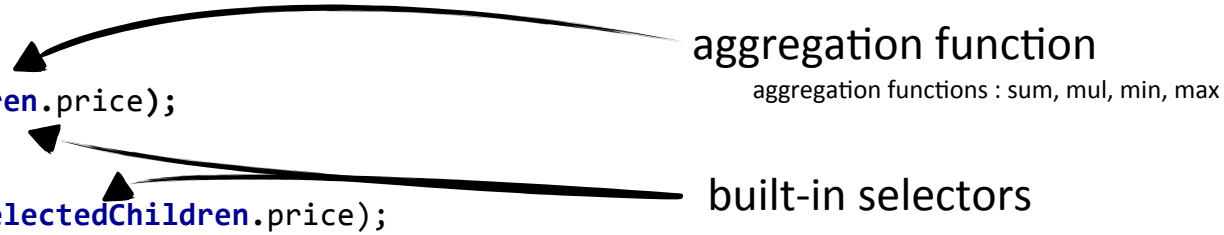
```
const int minSize is 17;
const int maxSize is 19;

Exterior {
  group allof {
      Wheels {
           int size in [minSize..maxSize]
      }
  }
}
```

constants can be reused

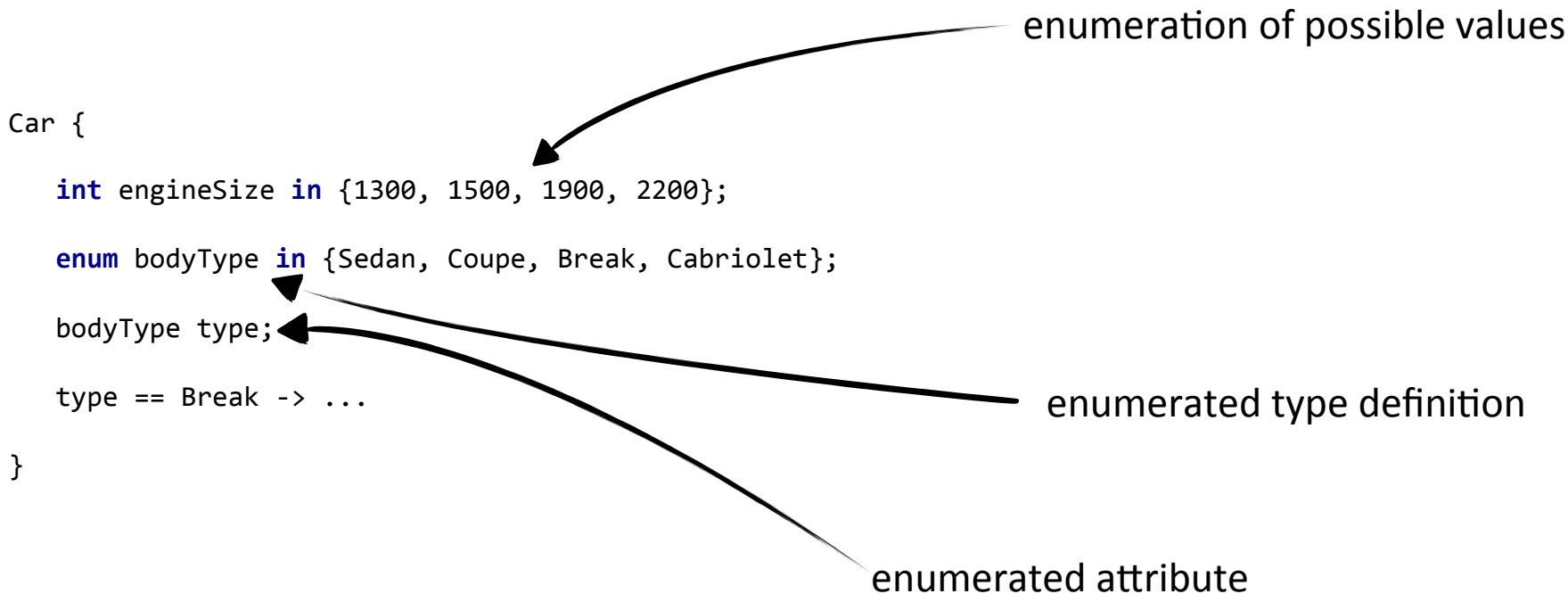
constants are allowed in cardinalities, expressions, etc.

TVL : Attributes

```
Car {
    int price is sum(children.price);

    Equipment {
        int price is sum(selectedChildren.price);

        group someof {
           NavigationSystems {
               int price is 605;
           },
           Telephone {
               int price is 300;
           },
           Speakers {
               int price is 690;
           },
           Other {
               int price is 385;
           }

        }
}
```



TVL : Attributes



TVL : ifIn / ifOut

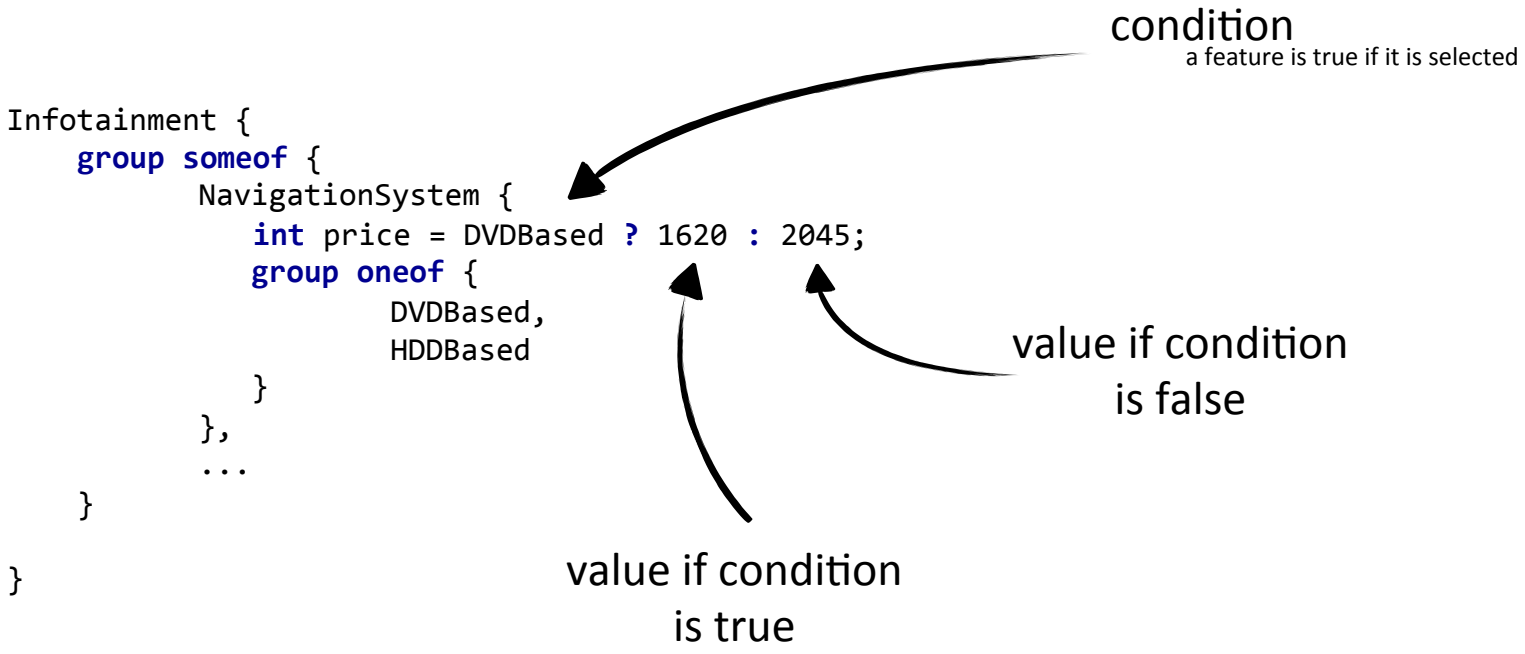
```
Infotainment {
     group someof {
          NavigationSystem {
             int price;
             group oneof {
                DVDBased {
                     ifIn: NavigationSystem.price == 1620;
                },
                HDDBased {
                     ifIn: NavigationSystem.price == 2045;
                }
             }
          },
          ...
    }

}
```

conditional constraint

ifIn : applies only if the feature is selected

ifOut : applies if the feature is not selected

TVL : Ternary operator



TVL : Modularity

```
root Car {
          group allOf {
          BodyStyle,
          Model,
          Engine,
          Equipment }

}

include(BodyStyle.tvl);
include(Model.tvl);
include(Engine.tvl);
include(Equipment.tvl);
```

Split model in reusable parts using external files

TVL : Modularity

BodyStyle.tvl

```
BodyStyle {
    group oneof {
           Avant,
           Door3,
           Sportback,
           Coupe,
           Cabriolet,
           Roadster,
           Spyder,
           SUV
     }

}
```

Model.tvl

```
Model {
    group oneof {
           A1,
           A3,
           S3,
           A4,
           S4,
           A5,
           S5,
           RS5,
           A6,
           A7,
           A8,
           A8L,
           R8
           Q3,
           Q5,
           Q7,
           TT,
           TTS,
           TT RS
     }
}
```

Engine.tvl

```
Engine {
   group oneof {
      SE12TFSI,
      CE14TFSI,
      CE14TFSI119,
      Sport12TFSI,
      Sport14TFSI,
      Sport14TFSI119,
      BlackEdition14TFSI,
      SLine12TFSI,
      SLine14TFSI,
      SE16TDI,
      CE16TDI,
      CE20TDI,
      ...
     }

}
```

TVL : Tools

- Key references
  - Classen, A.; Boucher, Q. and Heymans, P. A Text-based Approach to Feature Modelling: Syntax and Semantics of TVL. In Science of Computer Programming (SCP), Special Issue on Software Evolution
  - Hubaux, A.; Boucher, Q.; Hartman, H.; Michel, R. and Heymans, P. Evaluating a Text-based Feature Modelling Language: Four Industrial Case (SLE 2010)
  - http://info.fundp.ac.be/tvl/
- Support
  - Parser / Syntactic and semantic checker
  - SAT-based checker for Boolean models
- Ongoing
  - SMT-based checker for models with numerical attributes (ongoing)
  - Eclipse editor (ongoing)

2. Managing variability models with FAMILIAR

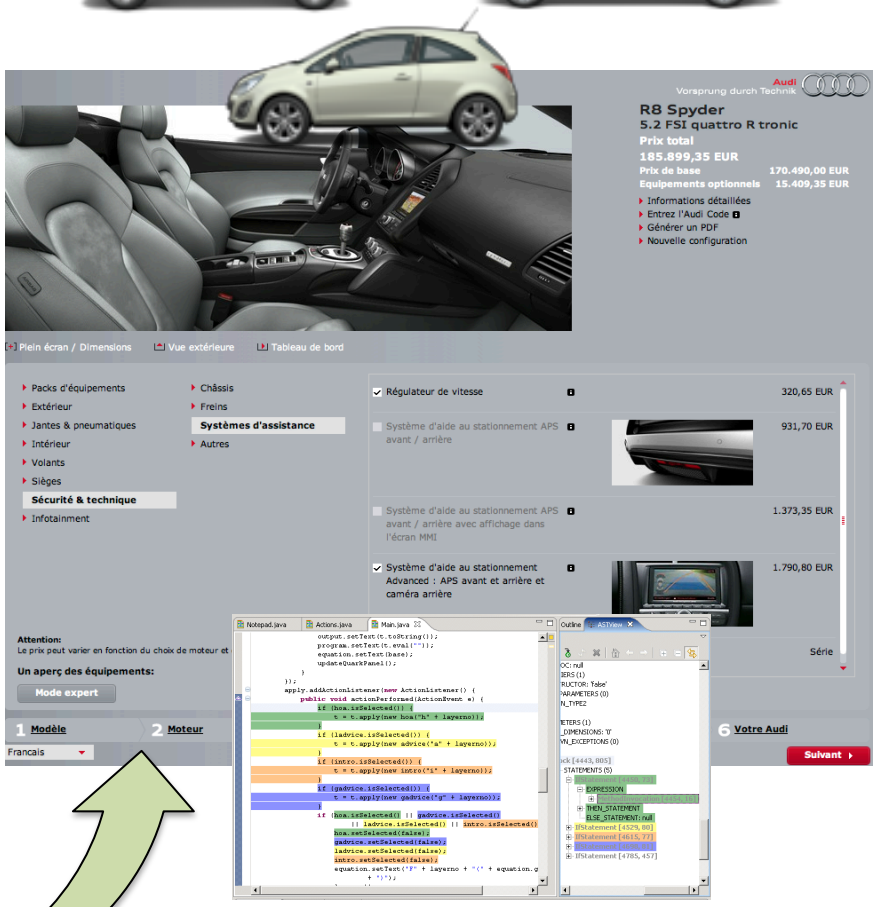
At the end of the tutorial (bis)

- You will have an overview of what's going on in the field of variability and model-based software product line engineering
- You will be able to go further with the languages and modelling techniques
  - reuse them in practical or academic contexts

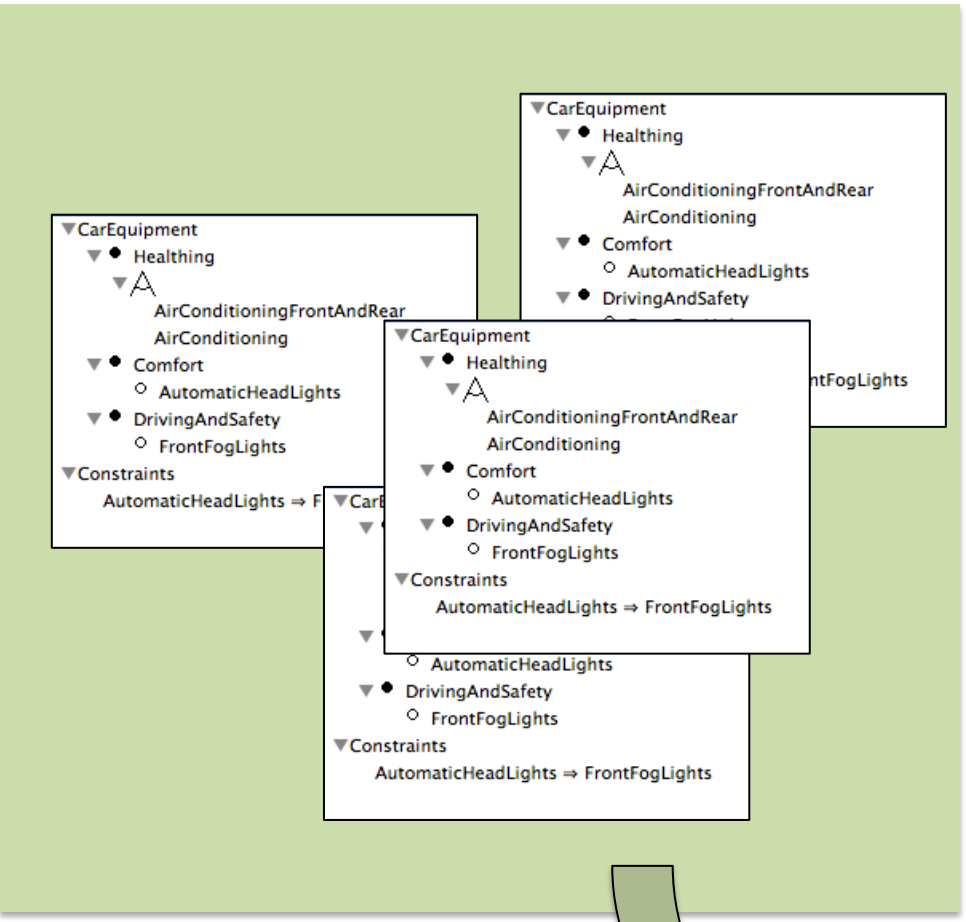
- Supporting material: https://nyx.unice.fr/projects/familiar/wiki/SPLC12-tutorial
  - slides of the tutorial
  - related articles,
  - TVL models,
  - FAMILIAR scripts,
  - and packaged tools to interactively play with the models during the tutorial.

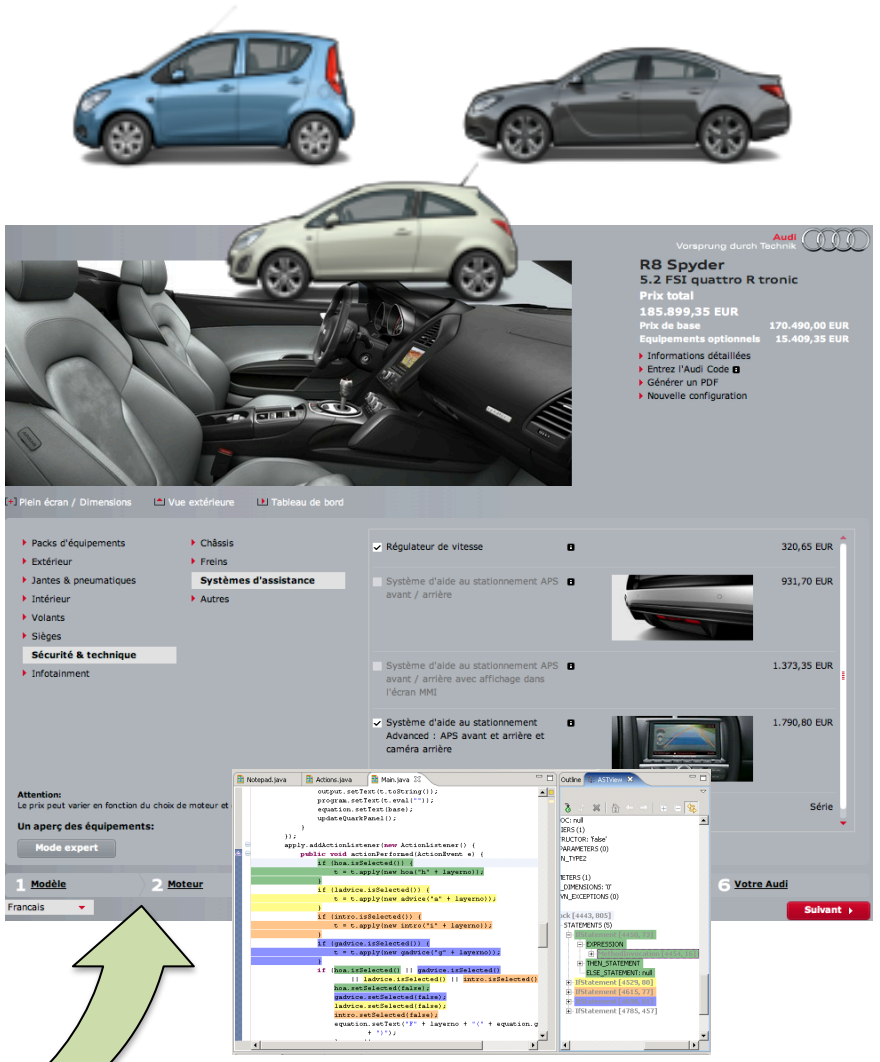
#1 Automated Analysis

- CarEquipment
  - Healthing
    - AirConditioningFrontAndRear
    - AirConditioning
  - Comfort
    - AutomaticHeadLights
  - DrivingAndSafety
    - FrontFogLights
- Constraints
  - AutomaticHeadLights ⇒ FrontFogLights

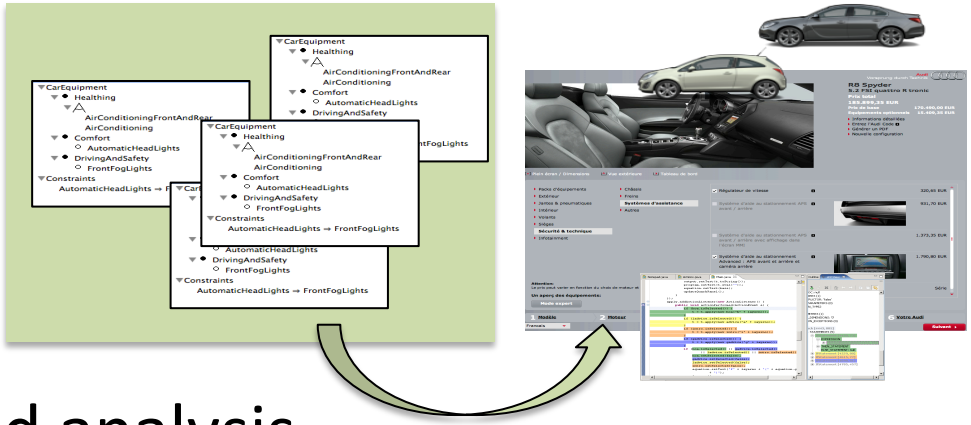


#2 Multiple Feature Models





Two Key Requirements



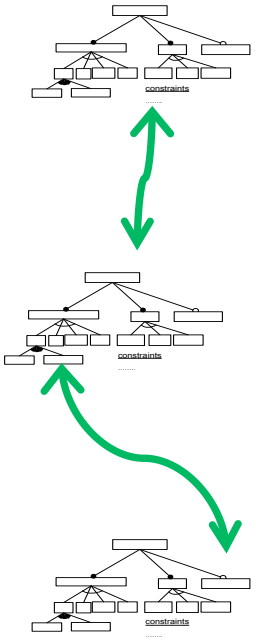
- #1 Automated analysis
  - Aka support to better understand and play with your feature model (TVL model)

Automated analysis of feature models 20 years later: A literature review ☆

David Benavides *, Sergio Segura, Antonio Ruiz-Cortés

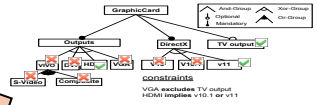
- #2 Managing <u>multiple</u> feature models
  - Composing / Decomposing / Diff and Reasoning about their relationships
  - Combining these operators

FAMILIAR language and environment

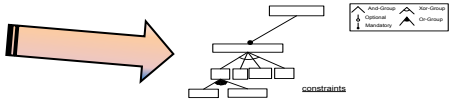



```
// foo.fml
fm1 = FM ("foo1.tvl")
fm2 = FM ("foo2.m")
fm3 = merge intersection { fm1 fm2 }
c3 = counting fm3
renameFeature fm3.TV as "OutputTV"
fm5 = aggregate { fm3 FM ("foo4.xml") }
assert (isValid fm5)
fm6 = slice fm5 including fm5.TV.*
export fm6
```

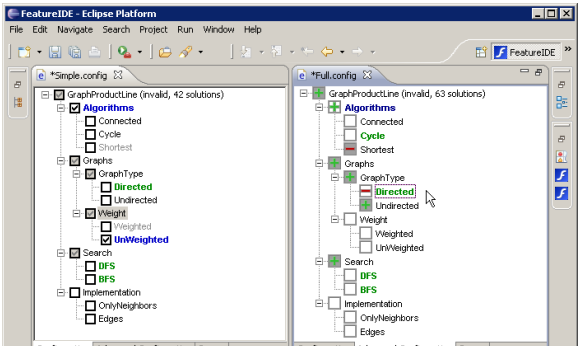
FAMILIAR

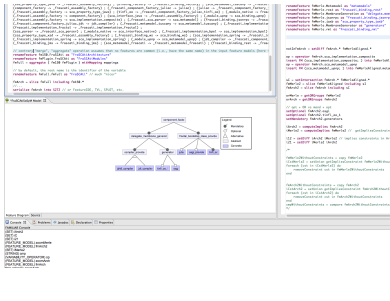


True/False
8759
"OutputTV", "TV"



Interoperability **Language facilities** **Environment**



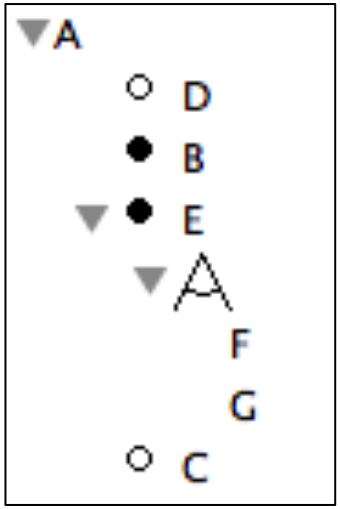


FAMILIAR … features

helloworld.fml

Hello World

```
fm1 = FM ("fool.tvl")
s = "hello world"
c = counting fm1
n = size fm1.*
println s
```




```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.6.jar helloworld.fml
hello world

FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)  version 0.9.9.6 (beta)
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> ls
(DOUBLE) c
(FEATURE_MODEL) fm1
(INTEGER) n
(STRING) s
fml>
```

basics2.fml

Typed language

- Domain-specific types
  - Feature Model,
  - Configuration,
  - Feature,
  - Constraint
- Other types include
  - Set
  - String
  - Boolean,
  - Enum,
  - Integer and Real.

```
fm1 = FM ("foo1.tvl")
ft1 = root fm1
ft2 = fm1.B
fts = fm1.*
n = size fts
n2 = cores fm1

cf = configuration fm1
select B in cf
deselect C in cf

cst1 = constraint (B -> !C)
addConstraint cst1 to fm1
```

- A set of operations, called **operators**, are defined for a given type.

basics2.fml

Typed language

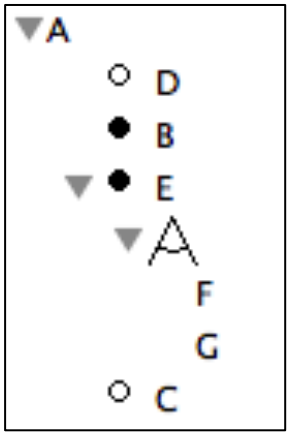
```
fm1 = FM ("foo1.tvl")
ft1 = root fm1
ft2 = fm1.B
fts = fm1.*
n = size fts
n2 = cores fm1

cf = configuration fm1
select B in cf
deselect C in cf

cst1 = constraint (B -> !C)
addConstraint cst1 to fm1
```

```
fml> ls
(FEATURE_MODEL) fm1
(SET) fts
(FEATURE) ft2
(CONSTRAINT) cst1
(INTEGER) n
(CONFIGURATION) cf
(SET) n2
(FEATURE) ft1
```

basics2.fml

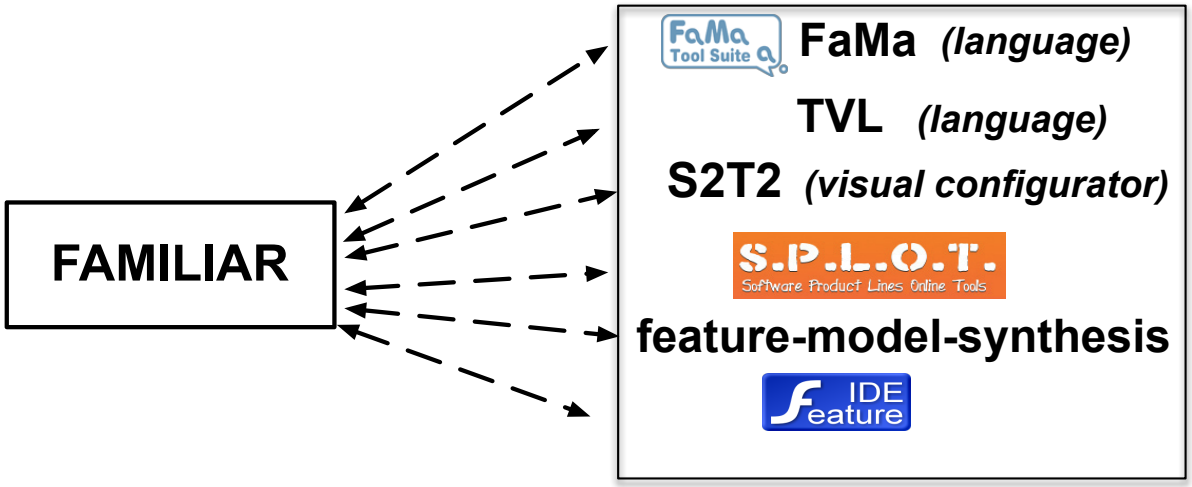
Typed language

```
fm1 = FM ("foo1.tvl")
ft1 = root fm1
ft2 = fm1.B
fts = fm1.*
n = size fts
n2 = cores fm1

cf = configuration fm1
select B in cf
deselect C in cf

cst1 = constraint (B -> !C)
addConstraint cst1 to fm1
```




```
fml> ft1
ft1: (FEATURE) A
fml> ft2
ft2: (FEATURE) B
fml> fts
fts: (SET) {G;E;D;F;C;A;B}
fml> n
n: (INTEGER) 7
fml> n2
n2: (SET) {E;A;B}
fml> cf
cf: (CONFIGURATION) selected: [E, A, B]          deselected: [C]
fml> cst1
cst1: (CONSTRAINT) (B -> !C)
fml> fm1
fm1: (FEATURE_MODEL) A: [D] B E [C] ;
E: (F|G) ;
(B -> !C);
```

Importing/Exporting feature models



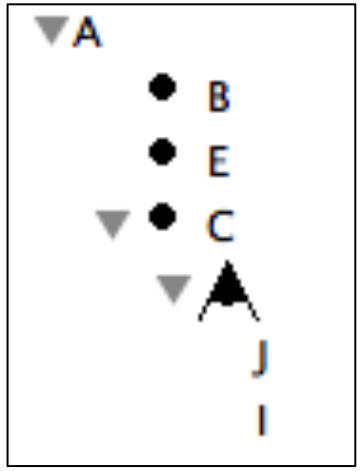
Internal notation *or* by "filename extensions"

```
fm1 = FM ("foo1.tvl")
fm2 = FM (A : [B] [C] D ; )
fm3 = FM ("foo2.m")
serialize fm2 into SPLOT // export
```

ftAccessors.fml

Feature Accessors (1)

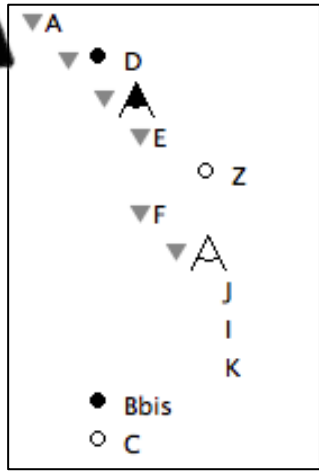
```
fm1 = FM (A : B [C] ; B : E F ; C : (I|J) ; )

r1 = root fm1
s = children r1
s1 = children fm1.A
assert (s eq s1) // equality of the two sets
ft1 = parent fm1.F
str1 = name ft1
ft2 = parent F // parent fm1.F

// another FM
fm2 = FM (A : B C E ; C : (I|J)+ ; )
ft3 = fm2.B
ft4 = name B // ambiguity
```



ftAccessors2.fml

Other constructs

```
fm1 = FM (A: B [C] D; D : (E|F)+; F : (I|J|K); E : [Z]; )
fm1bis = copy fm1 // save the original version

renameFeature fm1.B as "Bbis"
s1 = fm1.* // set of features of fm1
foreach (ft1 in s1) do
        println ft1
end
```

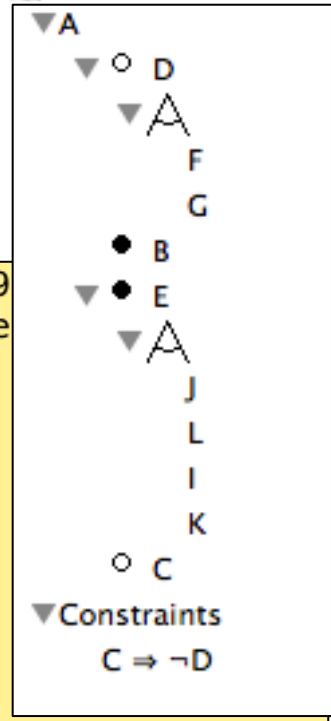



```
acher-scr:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar ftAccessors2.fml
Bbis

D

E

A

Z

I

C

J

K

F

FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)  version 0.9.9.5 (bet
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> exit
Bye, FAMILIAR user!
```

conf.fml

Configuration

```
fm1 = FM (A: B [C] [D] E; D : (F|G) ; E : (I|J|K|L) ; C -> !D ; )
c1 = configuration fm1
select C in c1
sc1 = selectedF c1 // accessors
cFM1 = asFM c1 // configuration and FM: back!
```

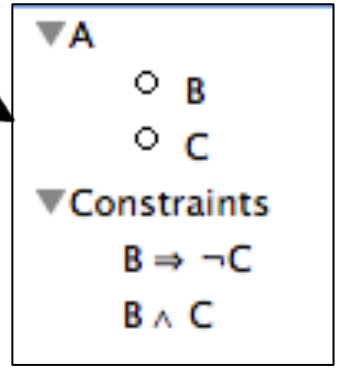



```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Re
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> cFM1
cFM1: (FEATURE_MODEL) A: B E C ;
E: (J|L|I|K) ;
E;
A;
B;
C;
fml> fm1
fm1: (FEATURE_MODEL) A: [D] B E [C] ;
D: (F|G) ;
E: (J|L|I|K) ;
(C -> !D);
```

operatorsFM.fml

Operations for Feature Models (1)

```
fm1 = FM (A : [B] [C] ; B -> !C ; B and C ; )
b1 = isValid fm1
```

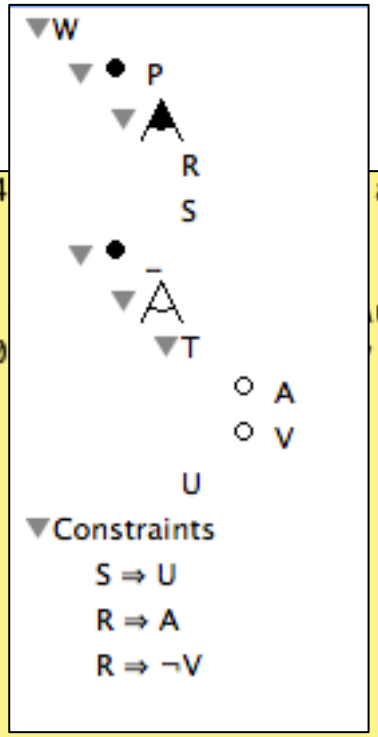



```
acher-scr:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar operatorsFM.f
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoni
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> ls
(FEATURE_MODEL) fm1
(BOOLEAN) b1
fml> b1
b1: (BOOLEAN) false
fml> configs fm1
res0: (SET) {}
```

operatorsFM2.fml

Operations for Feature Models (2)

```
1 fm1 = FM (W : P (T|U); P : (R|S)+ ; T : [V] [A] ; R -> !V ; S -> U ; R -> A ; )
2 b1 = isValid fm1
3 s1 = configs fm1
4 c1 = counting fm1
5 dfm1 = deads fm1
6 println "cores: ", cores fm1
7 fo1 = falseOptionals fm1
```



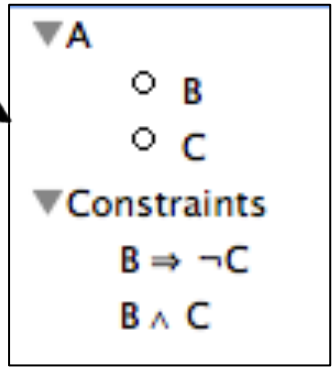

```
acher-scr:FML-scripts macher$ java -jar -Xmx1024                    ar operatorsFM2.fml
cores: {P;W}

FAMILIAR (for FeAture Model scrIpt Language for                     utomatic Reasoning)
University of Nice Sophia Antipolis, UMR CNRS 60
https://nyx.unice.fr/projects/familiar/
fml> ls
(SET) fo1
(SET) dfm1
(SET) s1
(DOUBLE) c1
(BOOLEAN) b1
(FEATURE_MODEL) fm1
fml> c1
c1: (DOUBLE) 2.0
fml> fo1
fo1: (SET) {A}
fml> dfm1
dfm1: (SET) {V}
```

φ

operatorsFM3.fml

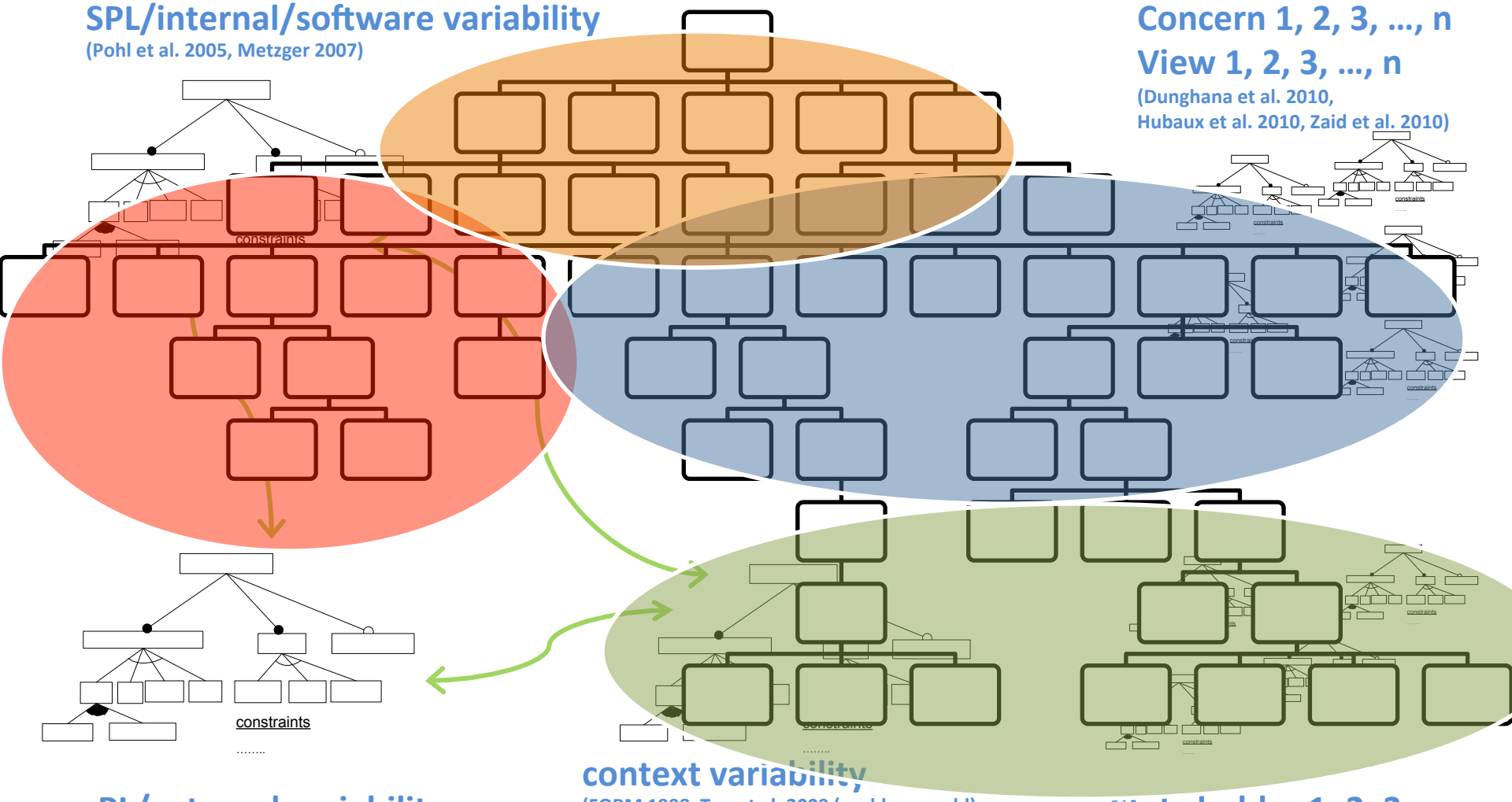
Operations for Feature Models (3)

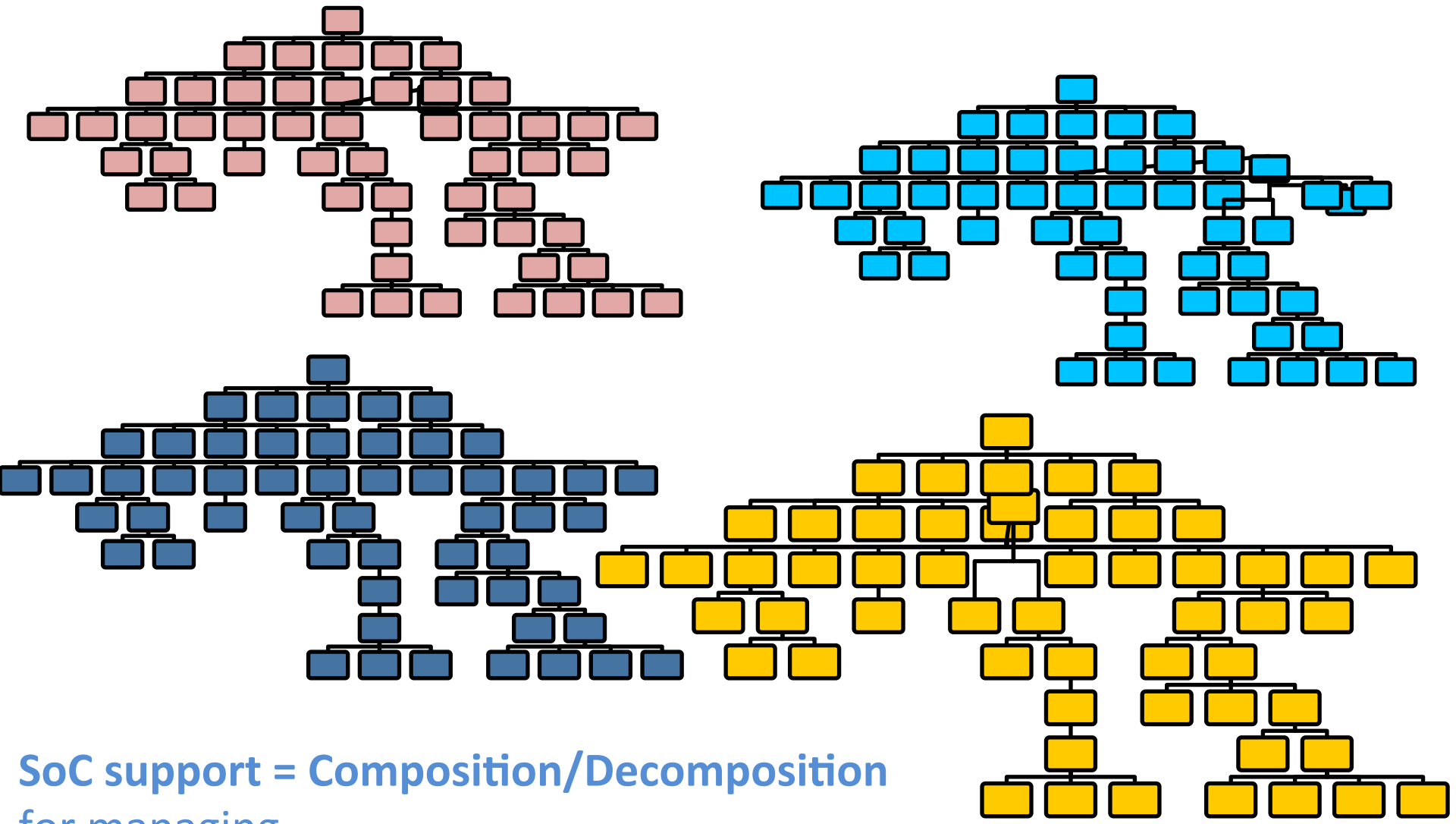
```
fm1 = FM (A : [B] [C] ; B -> !C ; B and C ; )
b1 = isValid fm1

csts1 = constraints fm1
foreach (cst in csts1) do
        println "removing constraint... ", cst
        removeConstraint cst in fm1
        c = counting fm1
        println "now the number of valid configurations is... ", c
end
```




```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar operatorsFM3.fml
removing constraint... (B & C)

now the number of valid configurations is... 3.0

removing constraint... (B -> !C)

now the number of valid configurations is... 4.0
```

Multiple Feature Models





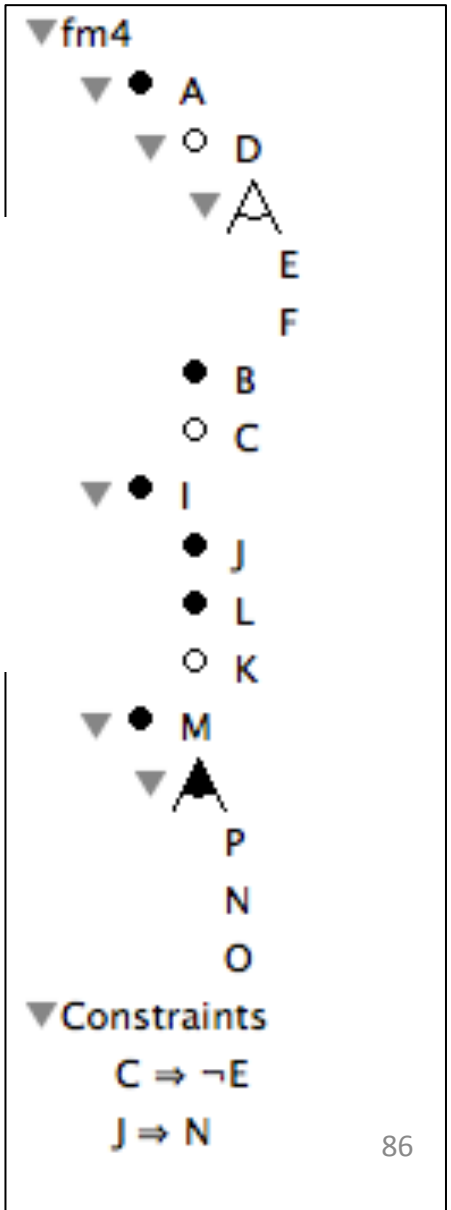
SoC support = Composition/Decomposition
for managing
large, complex and multiple
feature models

FORM 1998, Tun et al. 2009 (SPLC), Hartmann 2008 (SPLC), Lee et al. 2010, Czarnecki 2005, Reiser et al. 2007 (RE journal), Hartmann et al. 2009 (SPLC), Thuem et al. 2009 (ICSE), Classen et al. 2009 (SPLC), Mendonca et al. 2010 (SCP), Dunghana et al. 2010, Hubaux et al. 2011 (SoSyM), Zaid et al. 2010 (ER), She et al., 2011 (ICSE), etc.

aggregateBasics.fml

Composing Feature Models (1)

```
fm1 = FM (A : B [C] [D] ; D : (E|F) ; C -> !E; )
fm2 = FM (I : J [K] L ; )
fm3 = FM (M : (N|O|P)+ ; )
cst = constraints (J implies N ; )

// equivalent to aggregate { fm1 fm2 fm3 }
fm4 = aggregate fm* withMapping cst
```



aggregate1.fml

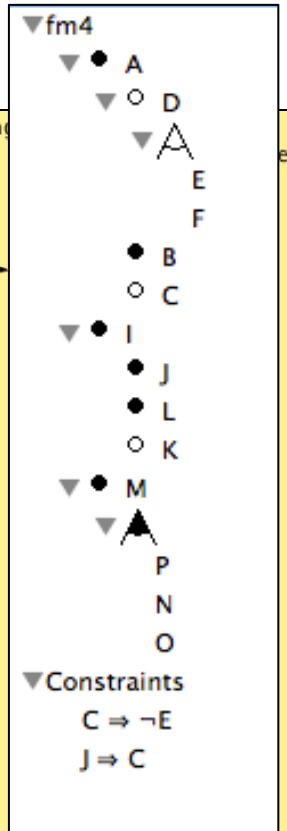
Composing Feature Models (2)

```
fm1 = FM (A : B [C] [D] ; D : (E|F) ; C -> !E; )
fm2 = FM (I : J [K] L ; )
fm3 = FM (M : (N|O|P)+ ; )
cst = constraints (J implies C ; )

// equivalent to aggregate { fm1 fm2 fm3 }
fm4 = aggregate fm* withMapping cst

// composition sometimes leads to "anomalies"
dfm4 = deads fm4
```

```
fm1 = FM (A : B [C] [D] ; D : (E|F) ; C -> !E; )
fm2 = FM (I : J [K] L ; )
fm3 = FM (M : (N|O|P)+ ; )
cst = constraints (J implies N ; )

// equivalent to aggregate { fm1 fm2 fm3 }
fm4 = aggregate fm* withMapping cst
```

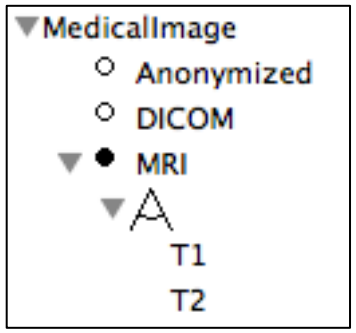
Previous version

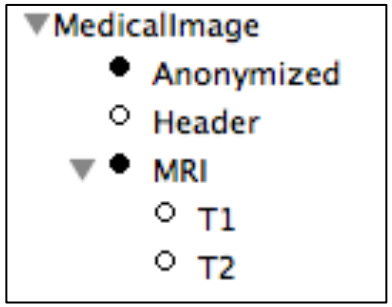
```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar ag
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> cores fm4
res0: (SET) {C;fm4;A;J;I;B;L;M}
fml> falseOptionals fm4
res1: (SET) {F;C}
fml> operator fm4.C
res2: (VARIABILITY_OPERATOR) OPTIONAL
fml> operator fm4.F
res3: (VARIABILITY_OPERATOR) ALTERNATIVE
fml> sibling fm4.F
res4: (SET) {E}
fml> deads fm4
res5: (SET) {E}
fml> operator fm4.E
res6: (VARIABILITY_OPERATOR) ALTERNATIVE
fml> fm4
fm4: (FEATURE_MODEL) fm4: A I M ;
A: [D] B [C] ;
I: J L [K] ;
M: (P|N|O)+ ;
D: (E|F) ;
(C -> !E);
(J -> C);
```

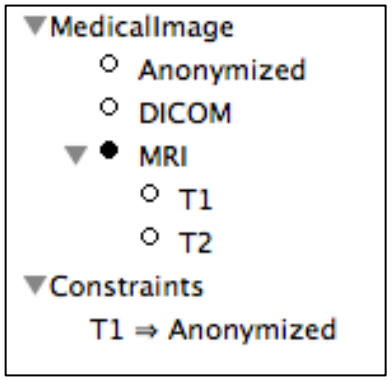


mergeMI.fml

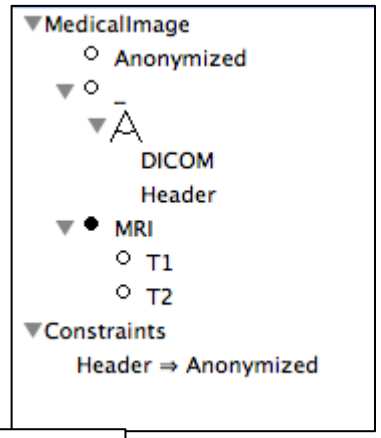
Merging Feature Models

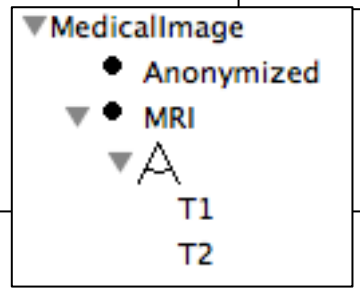







```
fmsupp1 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : (T1|T2) ;  )
fmsupp2 = FM (MedicalImage : Anonymized MRI [Header] ; MRI : [T1] [T2] ; )
fmsupp3 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : [T1] [T2] ; T1 -> Anonymized; )

s1 = configs fmsupp1
s2 = configs fmsupp2
s3 = configs fmsupp3

s123 = setUnion s3 setUnion s1 s2

fmSupp = merge sunion fmsupp*

assert (size s123 eq counting fmSupp)

fmCommon = merge intersection { fmsupp1 fmsupp2 }
sC = configs fmCommon
sC2 = setIntersection s1 s2
```



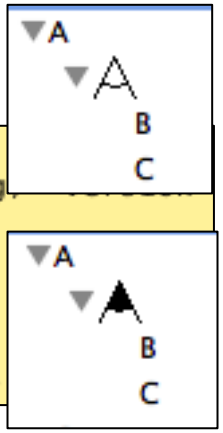


compare.fml

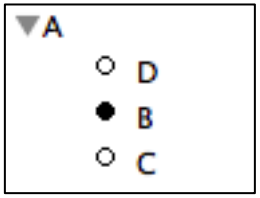
Comparing Feature Models

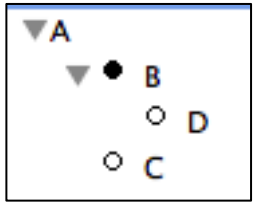
```
fm0 = FM (A: (B|C) ; )
fm1 = FM (A: (B|C)+ ; )
```

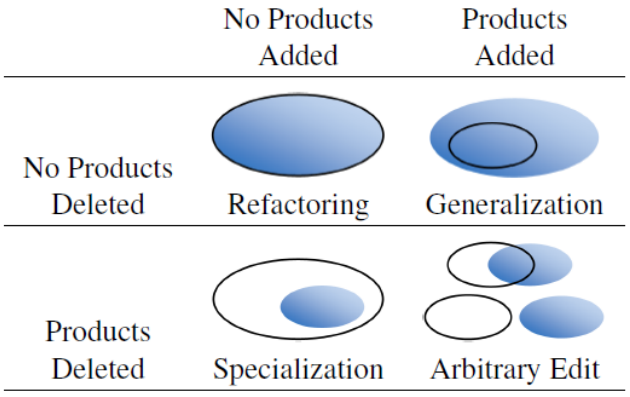
```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> cmp23
cmp23: (STRING) REFACTORING
fml>
```




```
assert (cmp10 eq GENERALIZATION)

// example taken from Automated Analysis of Feature Models
fm2 = FM (A: B [C] [D]; )
fm3 = FM (A: B [C]; B : [D]; )
cmp23 = compare fm2 fm3
```







see also Thuem, Kastner and Batory, ICSE'09

Putting all together: Example 1

Merge Intersection: Available Suppliers

- Registration
  - Affine
  - Header
    - A
      - v3
      - v1
  - Modality
    - CT
  - Format
    - A
      - DICOM
      - Nifti

- Registration
  - Affine
  - Header
    - A
      - v2
      - v1
  - Modality
    - MRI
  - Format
    - A
      - Nifti
      - GE

- Registration
  - Rigid
  - Header
    - A
      - v3
      - v2
  - Modality
    - A
      - PET
      - MRI
  - Format
    - Nifti

Suppliers?
Products?

- Registration
  - Header
    - v1
  - Modality
    - CT
  - Format
    - A
      - DICOM
      - Nifti

⊕ ∩ ⊕

- Registration
  - Header
    - v1
  - Modality
    - A
      - CT
      - MRI
  - Format
    - A
      - DICOM
      - Nifti
- Constraints
  - DICOM ⇒ ¬MRI

A customer has some requirements

suppliersExample0.fml

In FAMILIAR

```
REGsupp1 = FM (Registration : Header Format Modality [Affine] ;
                                                    Header : (v1|v3);
                                                    Format : (DICOM|Nifti) ;
                                                    Modality : CT; )

REGsupp2 = FM (Registration : Header [Affine] Format Modality ;
                                                    Header : (v1|v2);
```

```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.6.jar suppliersExample0.fml
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)  version 0.9.9.6 (beta)
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> ls
(FEATURE_MODEL) REGsupp3
(FEATURE_MODEL) REGsupp2p
(FEATURE_MODEL) REGrequired
(FEATURE_MODEL) REGsupp3p
(FEATURE_MODEL) REGsupp1p
(FEATURE_MODEL) REGsupp2
(FEATURE_MODEL) REGsupp1
fml> REGsupp3p
REGsupp3p: (FEATURE_MODEL) False
fml> REGsupp1p
REGsupp1p: (FEATURE_MODEL) Registration: Header Modality Format ;
Header: v1 ;
Modality: CT ;
Format: (DICOM|Nifti) ;
```

```
REGsupp1p = merge intersection { REGrequired REGsupp1 }
REGsupp2p = merge intersection { REGrequired REGsupp2 }
REGsupp3p = merge intersection { REGrequired REGsupp3 }
```

Merge Union: Availability Checking

Registration
- Affine
- Header
  - v3
  - v1
  - odality
  - CT
  - rmat
  - DICOM
  - Nifti

Registration
- Affine
- Header
  - v2
  - v1
- Modality
  - MRI
- Format
  - Nifti
  - GE

Registration
- Rigid
- Header
  - v3
  - v2
- Modality
  - PET
  - MRI
- Format
  - Nifti

Registration
- Header
  - v3
  - v2
  - v1
- Modality
  - PET
  - CT
  - MRI
- _
  - Affine
  - Rigid
- Format
  - DICOM
  - Nifti
  - GE

Constraints
- GE ⇒ MRI
- GE ⇒ ¬v3
- CT ⇒ ¬v2
- Rigid ⇒ Nifti
- DICOM ⇒ CT
- v1 ⇒ ¬Rigid
- CT ⇒ ¬Rigid
- v1 ⇒ ¬PET
- PET ⇒ ¬Affine

⊕U

Yes!

Can suppliers provide *all* products?

"compare"

Registration
- Header
  - v1
- Modality
  - CT
  - MRI
- Format
  - DICOM
  - Nifti

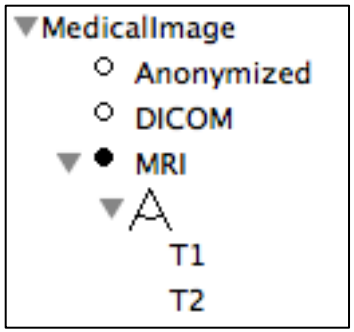
Constraints
- DICOM ⇒ ¬MRI

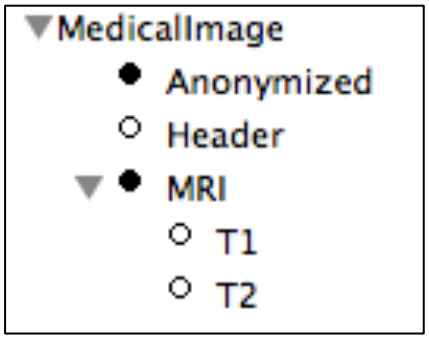
suppliersExample.fml

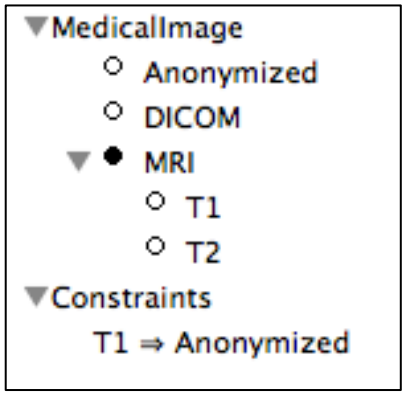
In FAMILIAR

```
REGsupp1 = FM (Registration : Header Format Modality [Affine] ;
                                                    Header : (v1|v3);
                                                    Format : (DICOM|Nifti) ;
                                                    Modality : CT; )

REGsupp2 = FM (Registration : Header [Affine] Format Modality ;
                                                    Header : (v1|v2);
                                                    Format : (Nifti|GE) ;
                                                    Modality : MRI; )

REGsupp3 = FM (Registration : Header [Rigid] Format Modality ;
                                                    Header : (v2|v3) ;
                                                    Format : Nifti ;
                                                    Modality : (MRI|PET); )


REGrequired = FM (Registration : Header Format Modality ;
                                                    Header : v1 ; //v3;
                                                    Format : (DICOM|Nifti) ;
                                                    Modality: (MRI|CT);
                                                    !DICOM or !MRI;
                                                    )

REGmspl = merge sunion REGsupp*          // merge all FMs whose variable identifier starts w

cmp = compare REGrequired REGmspl
//missingSPL = merge diff { REGrequired REGmspl }
```

Merging operation: implementation issues








```
fmsupp1 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : (T1|T2) ;  )
fmsupp2 = FM (MedicalImage : Anonymized MRI [Header] ; MRI : [T1] [T2] ; )
fmsupp3 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : [T1] [T2] ; T1 -> Anonymized; )

// computing the union of sets of configurations like this is COSTLY
s1 = configs fmsupp1
s2 = configs fmsupp2
s3 = configs fmsupp3

s123 = setUnion s3 setUnion s1 s2

// you WONT scale
//...

fmSupp = merge sunion fmsupp*

assert (size s123 eq counting fmSupp)
```

Anonymized v Header v DICOM v ~T1 v ~T2

Merging operation: semantic issues (2)

φ

$$s_0 = \{$$

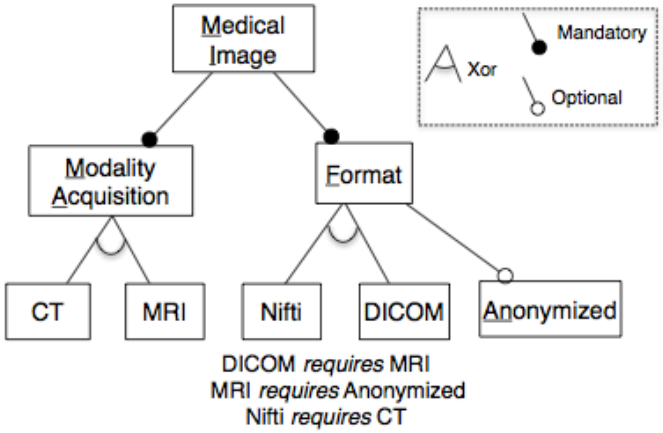
$$\{MI, MA, F, CT, Nifti\},$$

$$\{MI, MA, F, CT, Nifti, AN\},$$

$$\{MI, MA, F, DICOM, MRI, AN\}$$

$$\}$$

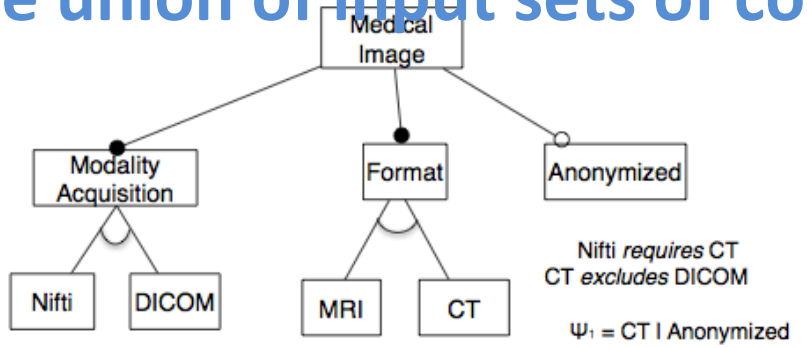
Union
Intersection
Diff



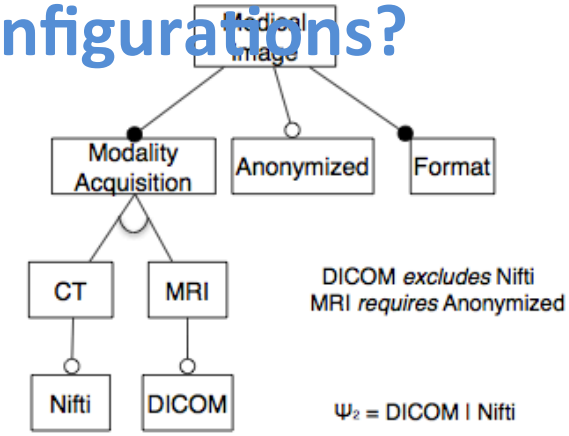
(a) set of configurations

(b) fm_0

How to synthesise a feature model that represents the union of input sets of configurations?



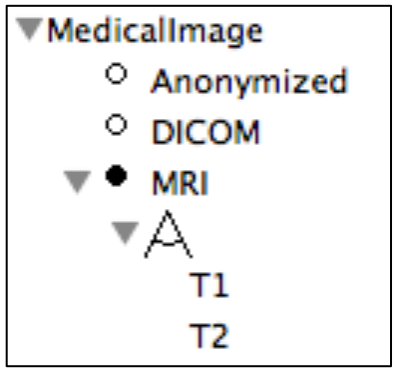
(c) fm_1



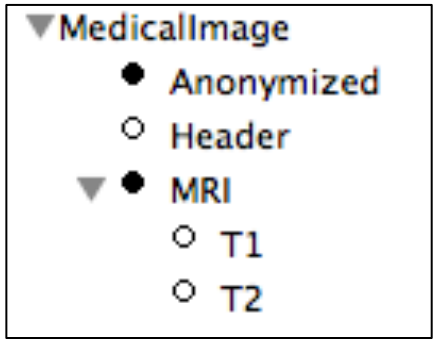
(d) fm_2

Fig. 2: For a given set of configurations, three possible yet different FMs ($s_0 = [\![fm_0]\!] = [\![fm_1]\!] = [\![fm_2]\!]$)

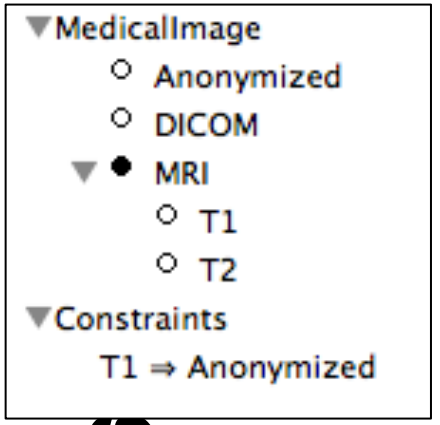
Merging operation: algorithm



φ_1



φ_2



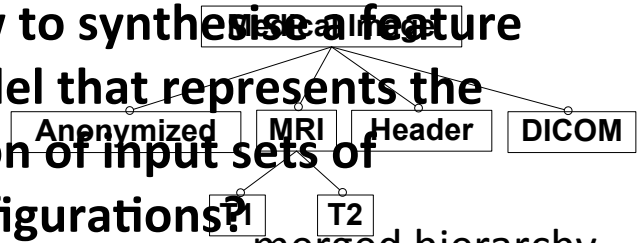
φ_3

φ_{123}

merged propositional formula

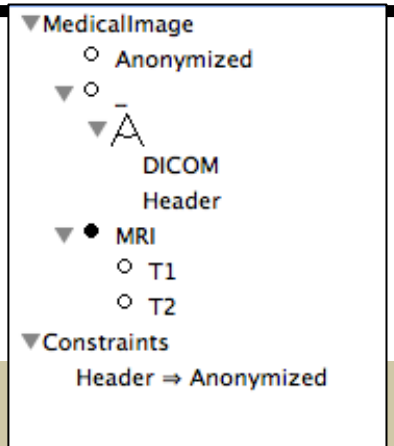
+

How to synthesise a feature model that represents the union of input sets of configurations?



merged hierarchy

Set mandatory features
Detect Xor and Or-groups
Compute "implies/excludes" constraints

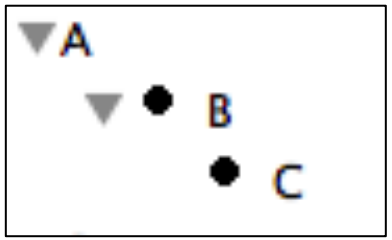


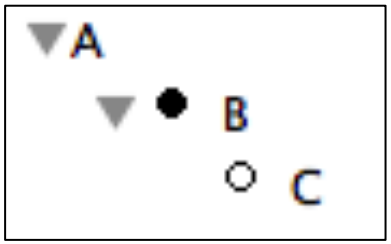
see also [Czarnecki SPLC'07 or SPLC'12]

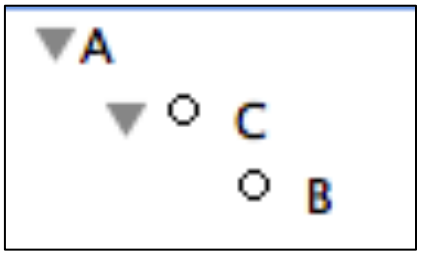
mergeNonPC.fml

Merging operation: back to hierarchy

```
fm1 = FM (A : B ; B : C ;)
fm2 = FM (A : B ; B : [C] ;)
fm3 = FM (A : [C] ; C : [B] ; )
fm4 = merge sunion { fm1 fm2 fm3 }
```







> configs fm4
res12: (SET) {{C;A};{A;B};{A};{A;B;C}}

?

see also [Acher PhD thesis]

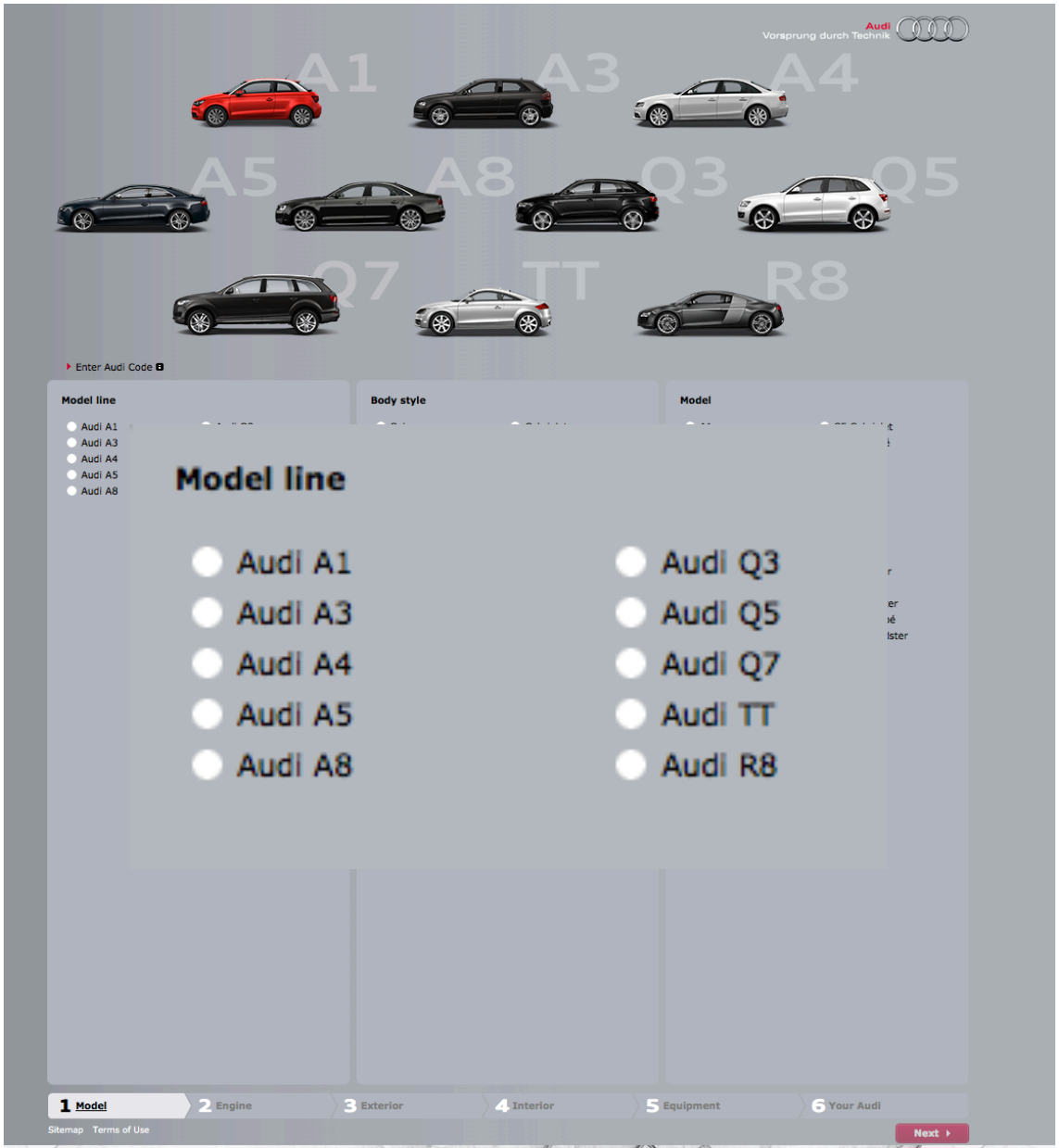
Related Works

- Well-defined semantics
- Guarantee semantics properties by construction
- More compact feature models than reference-based techniques [Schobbens et al., 2007], [Hartmann et al., 2007]
  - Easier to understand
  - Easier to analyze (e.g., compare with another)
- Applicable to any propositional feature models
  - Full support of propositional constraints
  - Different hierarchies [Van Den Broek et al., SPLC'2010/2012]
- Syntactical strategies fail [Alves et al., 2006], [Segura et al., 2007]

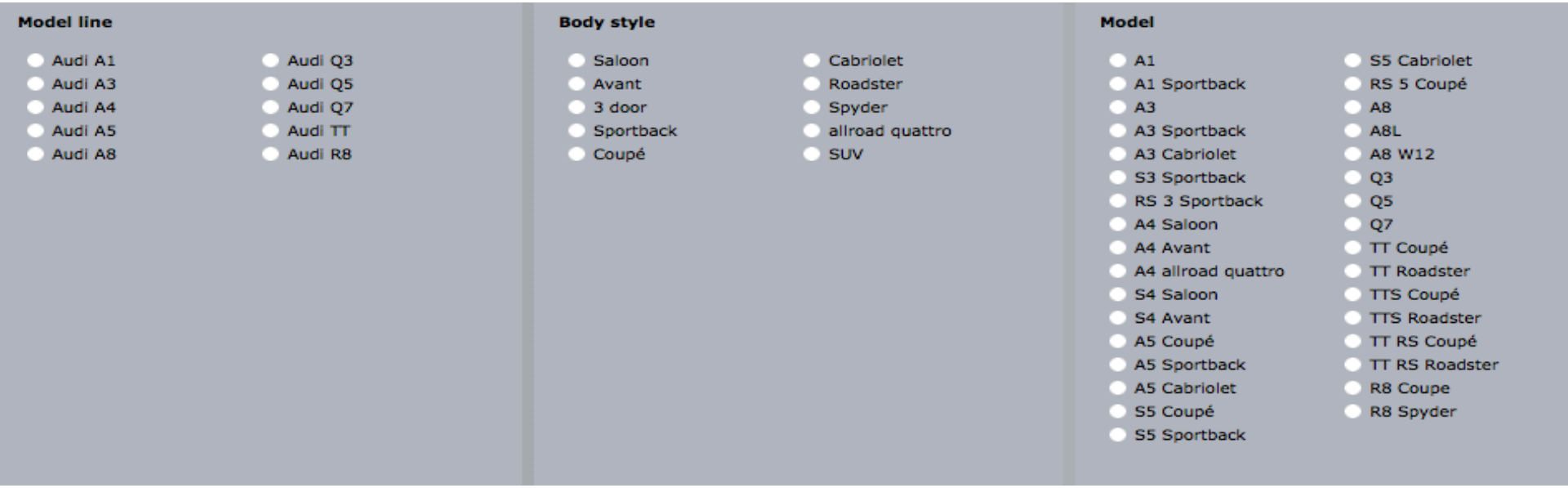
see also [Acher et al., ECMFA'12/AOSD'12/PhD thesis]

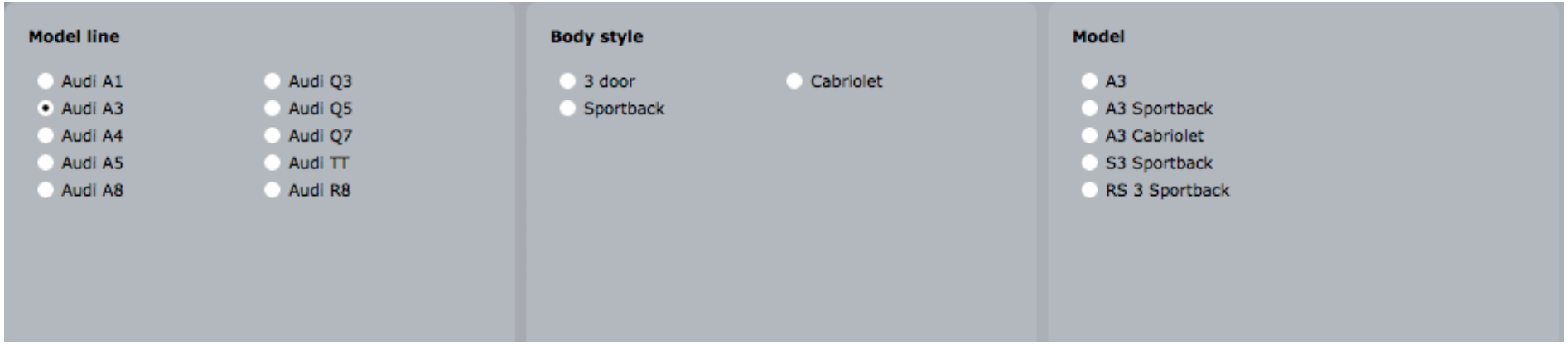
Another Example

Problem: multiple „car models“



Problem: multiple „car models“





Problem: multiple „car models“

Model line

- ○ Audi A1
- ◉ Audi A3
- ○ Audi A4
- ○ Audi A5
- ○ Audi A8
- ○ Audi Q3
- ○ Audi Q5
- ○ Audi Q7
- ○ Audi TT
- ○ Audi R8

Body style

- ○ 3 door
- ○ Sportback
- ○ Cabriolet

Model

- ○ A3
- ○ A3 Sportback
- ○ A3 Cabriolet
- ○ S3 Sportback
- ○ RS 3 Sportback

Model line

- ○ Audi A1
- ○ Audi A3
- ◉ Audi A4
- ○ Audi A5
- ○ Audi A8
- ○ Audi Q3
- ○ Audi Q5
- ○ Audi Q7
- ○ Audi TT
- ○ Audi R8

Body style

- ○ Saloon
- ○ Avant
- ○ allroad quattro

Model

- ○ A4 Saloon
- ○ A4 Avant
- ○ A4 allroad quattro
- ○ S4 Saloon
- ○ S4 Avant

Model line

- ◉ Audi A1
- ○ Audi A3
- ○ Audi A4
- ○ Audi A5
- ○ Audi A8
- ○ Audi Q3
- ○ Audi Q5
- ○ Audi Q7
- ○ Audi TT
- ○ Audi R8

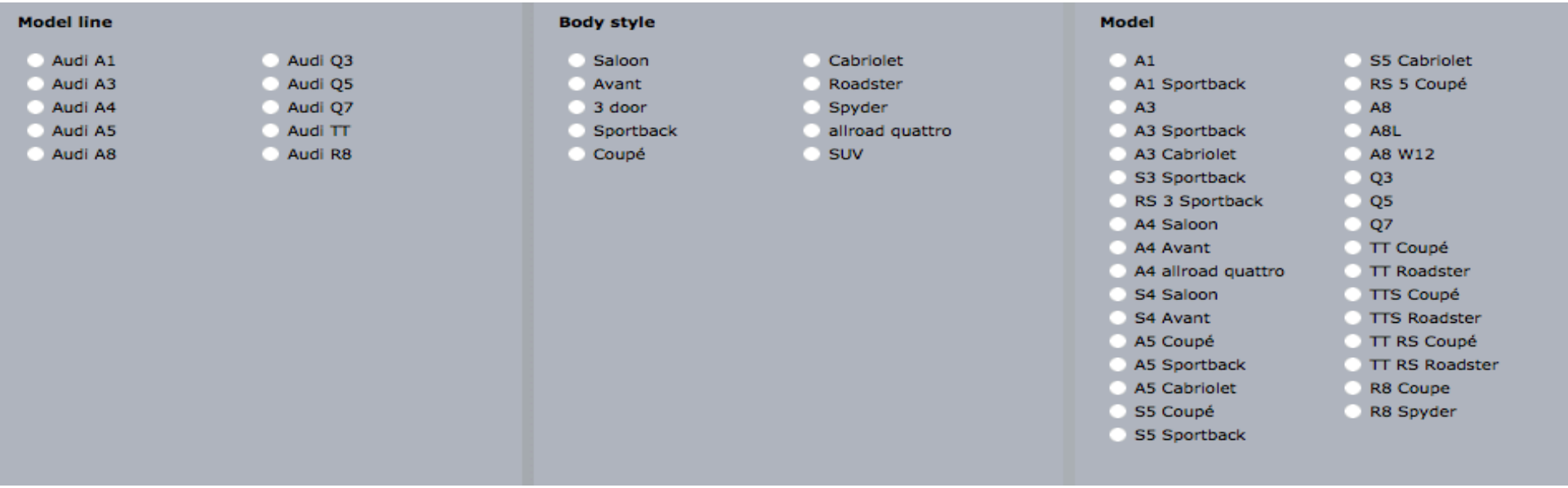
Body style

- ○ 3 door
- ○ Sportback

Model

- ○ A1
- ○ A1 Sportback

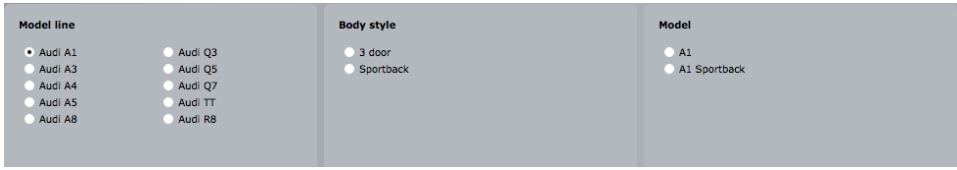
Problem: multiple „car models“



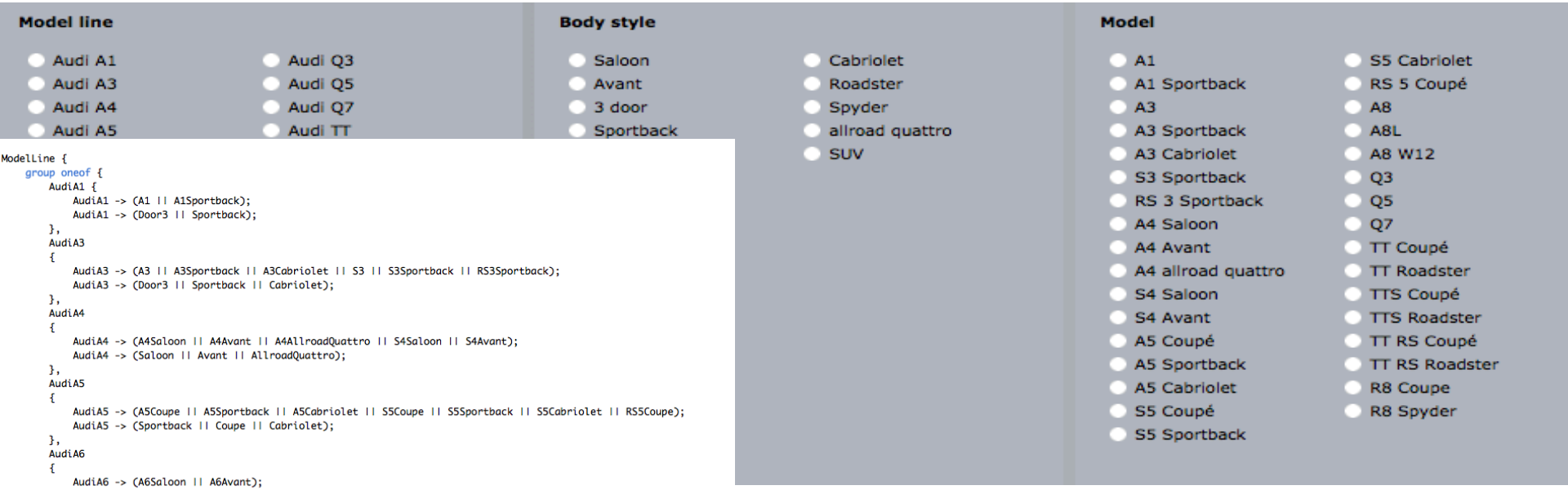
#1 – top-down: specify constraints (e.g., excludes) of all model lines upfront

Two modeling approaches

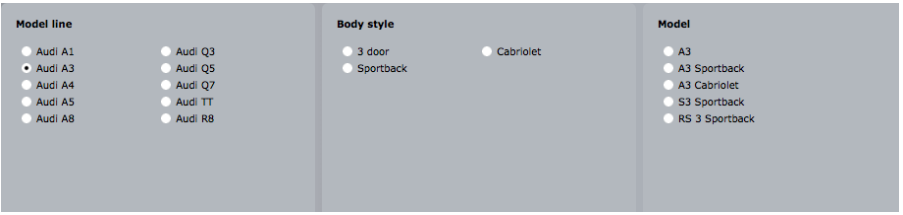
#2 – bottom-up: elaborate a feature model for each model line and merge them

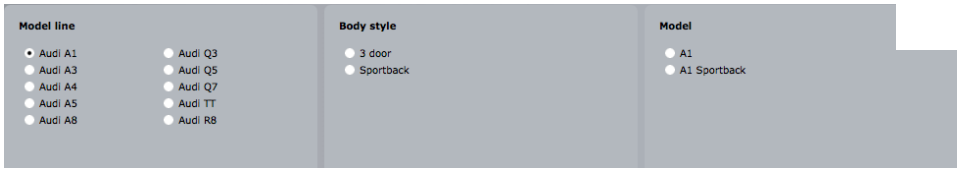


#1 top-down



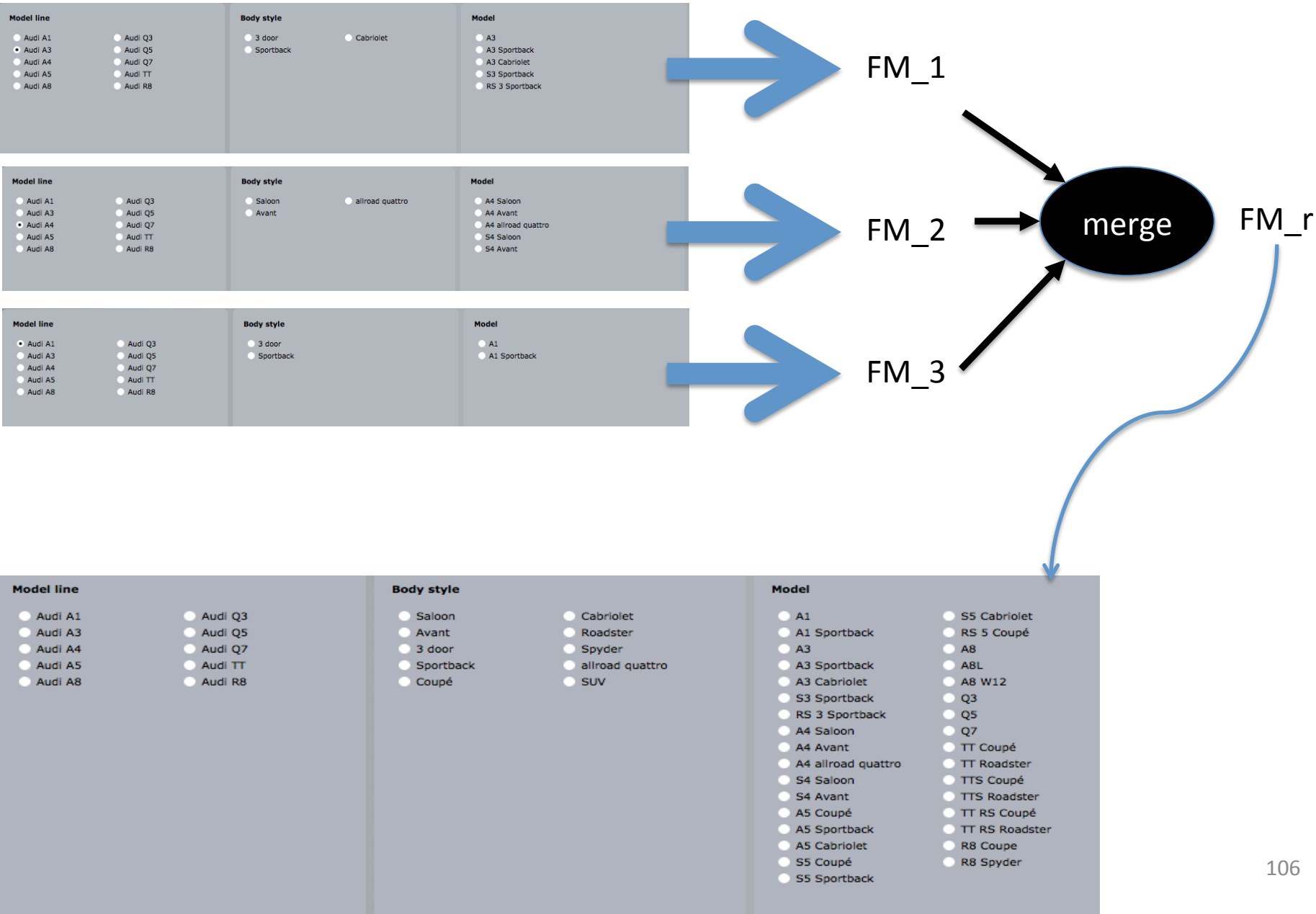

```
ModelLine {
    group oneof {
        AudiA1 {
            AudiA1 -> (A1 || A1Sportback);
            AudiA1 -> (Door3 || Sportback);
        },
        AudiA3
        {
            AudiA3 -> (A3 || A3Sportback || A3Cabriolet || S3 || S3Sportback || RS3Sportback);
            AudiA3 -> (Door3 || Sportback || Cabriolet);
        },
        AudiA4
        {
            AudiA4 -> (A4Saloon || A4Avant || A4AllroadQuattro || S4Saloon || S4Avant);
            AudiA4 -> (Saloon || Avant || AllroadQuattro);
        },
        AudiA5
        {
            AudiA5 -> (A5Coupe || A5Sportback || A5Cabriolet || S5Coupe || S5Sportback || S5Cabriolet || RS5Coupe);
            AudiA5 -> (Sportback || Coupe || Cabriolet);
        },
        AudiA6
        {
            AudiA6 -> (A6Saloon || A6Avant);
            AudiA6 -> (Saloon || Avant);
        },
```





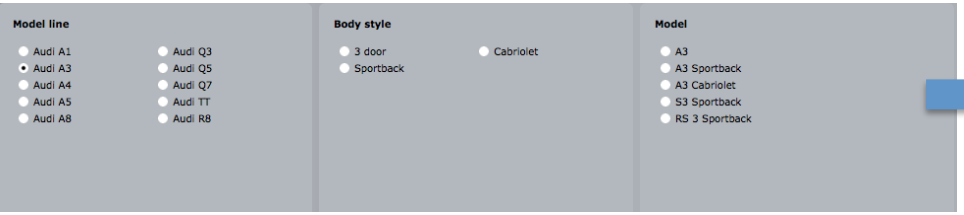

```
BodyStyle {
    group oneof {
        Saloon
        {
            Saloon -> (A4Saloon || S4Saloon || A6Saloon || A8 || A8L || A8W12);
        },
        Avant
        {
            Avant -> (A4Avant || S4Avant || A6Avant);
        },
        Door3
        {
            Door3 -> (A1 || A3 || S3);
        },
        Sportback
        {
            Sportback -> (A1Sportback || A3Sportback || S3Sportback || RS3Sportback || A5Sportback || S5Sportback || A7Spor
        },
        Coupe
        {
            Coupe -> (A5Coupe || S5Coupe || RS5Coupe || TTCoupe || TTSCoupe || TTRSCoupe || R8Coupe);
        },
```

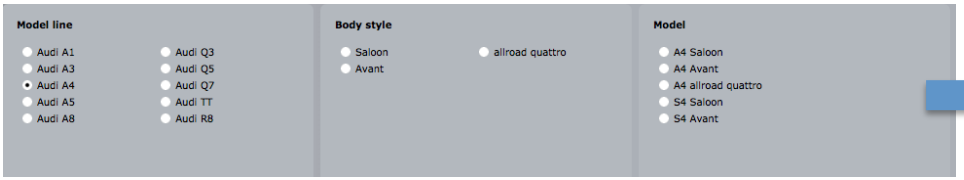
#1 bottom-up

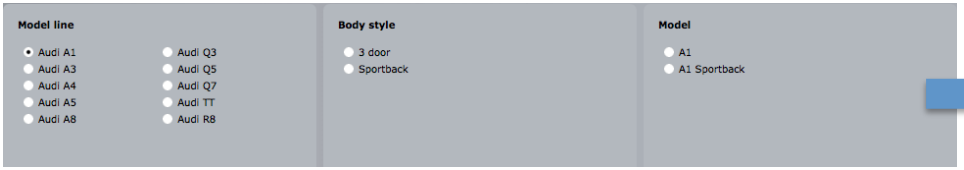

Model line
Audi A1
Audi A3
Audi A4
Audi A5
Audi A8
Audi Q3
Audi Q5
Audi Q7
Audi TT
Audi R8
Body style
3 door
Sportback
Cabriolet
Model
A3
A3 Sportback
A3 Cabriolet
S3 Sportback
RS 3 Sportback
FM_1
Model line
Audi A1
Audi A3
Audi A4
Audi A5
Audi A8
Audi Q3
Audi Q5
Audi Q7
Audi TT
Audi R8
Body style
Saloon
Avant
allroad quattro
Model
A4 Saloon
A4 Avant
A4 allroad quattro
S4 Saloon
S4 Avant
FM_2
Model line
Audi A1
Audi A3
Audi A4
Audi A5
Audi A8
Audi Q3
Audi Q5
Audi Q7
Audi TT
Audi R8
Body style
3 door
Sportback
Model
A1
A1 Sportback
FM_3
merge
FM_r
Model line
Audi A1
Audi A3
Audi A4
Audi A5
Audi A8
Audi Q3
Audi Q5
Audi Q7
Audi TT
Audi R8
Body style
Saloon
Avant
3 door
Sportback
Coupé
Cabriolet
Roadster
Spyder
allroad quattro
SUV
Model
A1
A1 Sportback
A3
A3 Sportback
A3 Cabriolet
S3 Sportback
RS 3 Sportback
A4 Saloon
A4 Avant
A4 allroad quattro
S4 Saloon
S4 Avant
A5 Coupé
A5 Sportback
A5 Cabriolet
S5 Coupé
S5 Sportback
S5 Cabriolet
RS 5 Coupé
A8
A8L
A8 W12
Q3
Q5
Q7
TT Coupé
TT Roadster
TTS Coupé
TTS Roadster
TT RS Coupé
TT RS Roadster
R8 Coupe
R8 Spyder

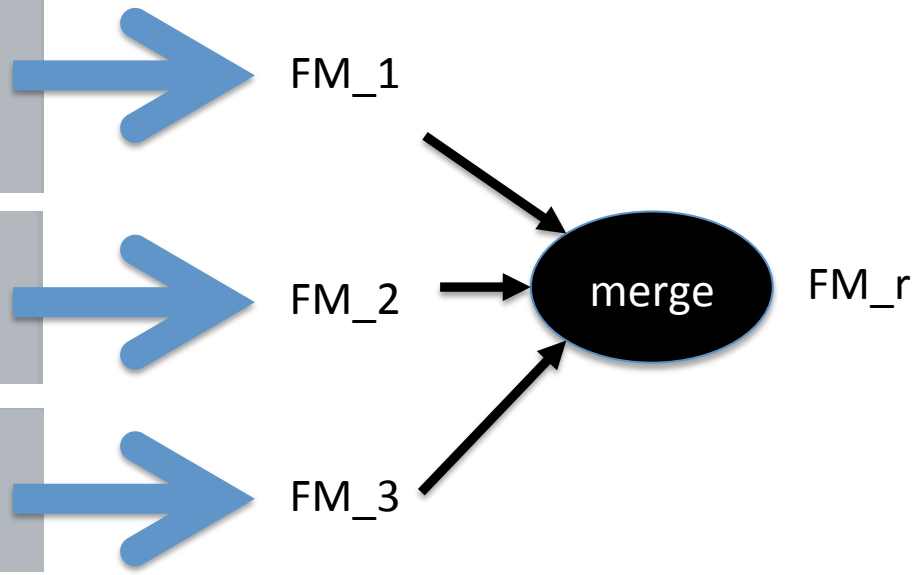
audiMerge.fml

#1 bottom-up (FAMILIAR)



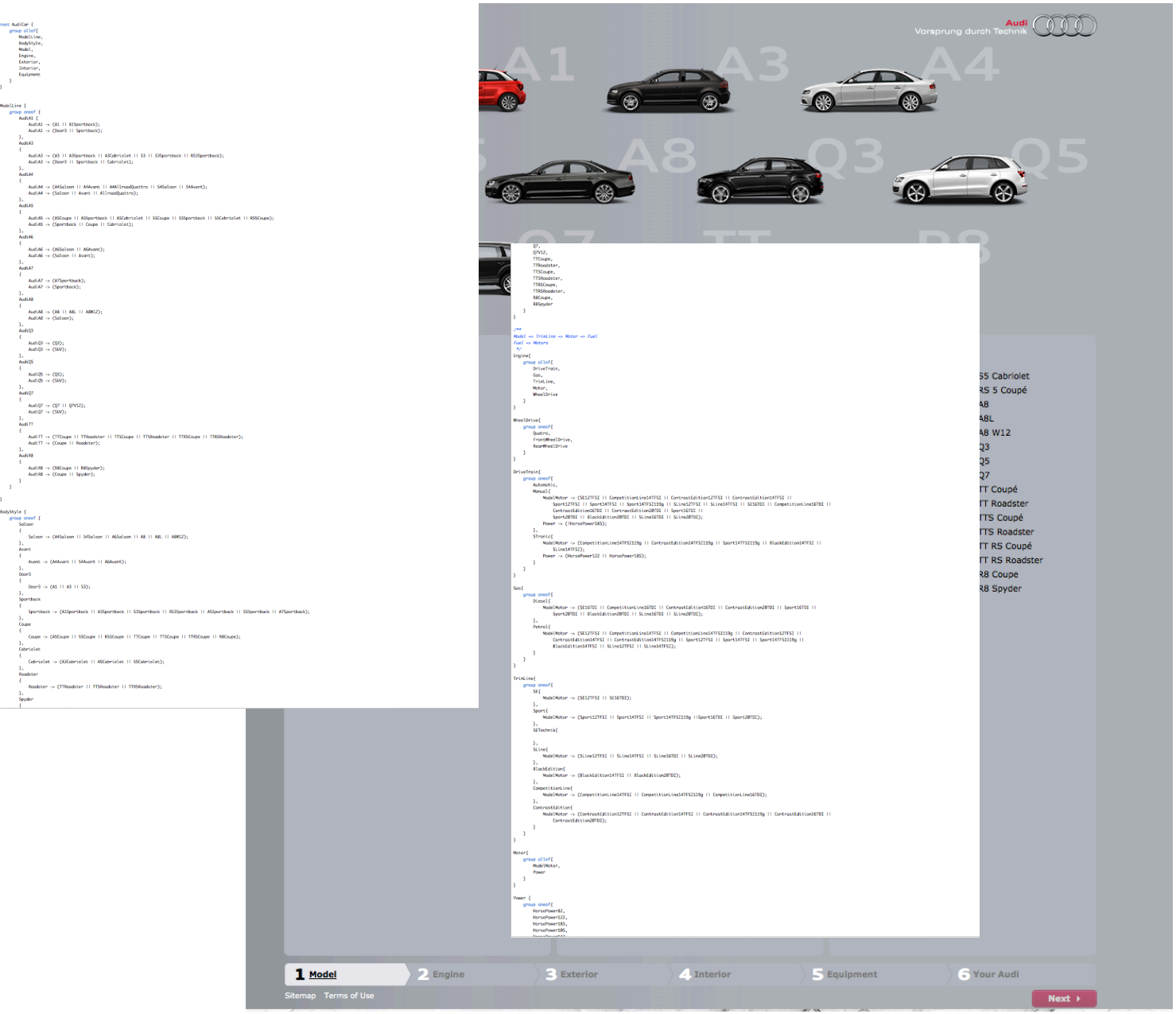







```
fml> fmAudiS = merge sunion { fm1 fm2 fm3 }
fmAudiS: (FEATURE_MODEL) Audi: ModelLine BodyStyle ;                    9.9.4 (beta)
ModelLine: (A1|A4|A3) ;
BodyStyle: (Saloon|Door3)? (Cabriolet|AllroadQuattro)? (Sportback|Avant)? ;
(A4 -> !Sportback);
(A1 -> !Avant);
(A1 -> !Saloon);
(A3 -> !Saloon);
(A4 -> !Cabriolet);
(A1 -> !AllroadQuattro);
(A1 -> !Cabriolet);
(A4 -> !Door3);
(A3 -> !Avant);
(A3 -> !AllroadQuattro);
```

Building “views” of a feature model

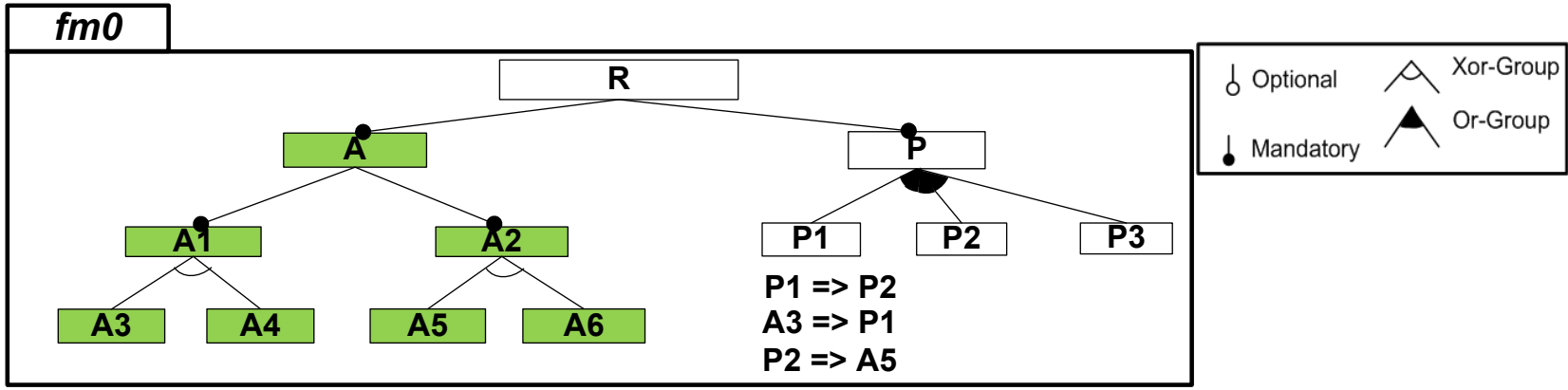


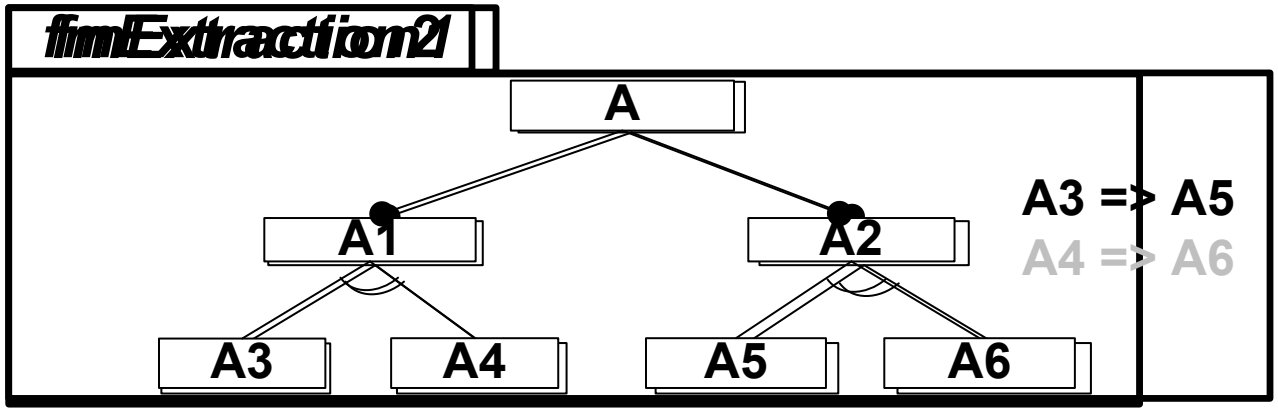
Building "views" of a feature model

- Problem: given a feature model, how to decompose it into smaller feature models?

- Semantics?
  - What's the hierarchy
  - What's the set of configurations?

A first try



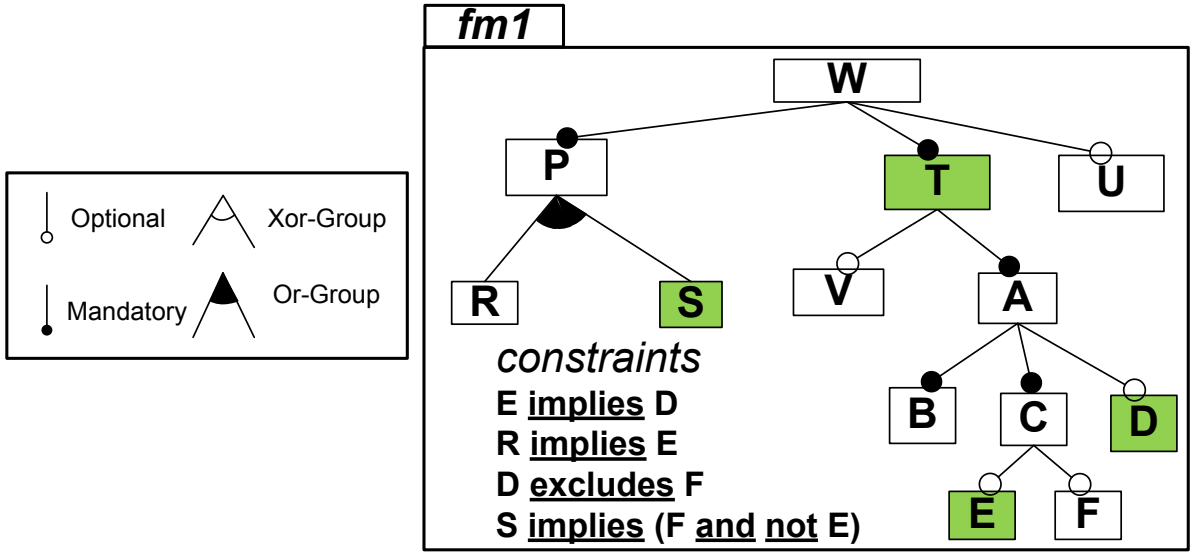


Problem: You can select **A3** without **A5**

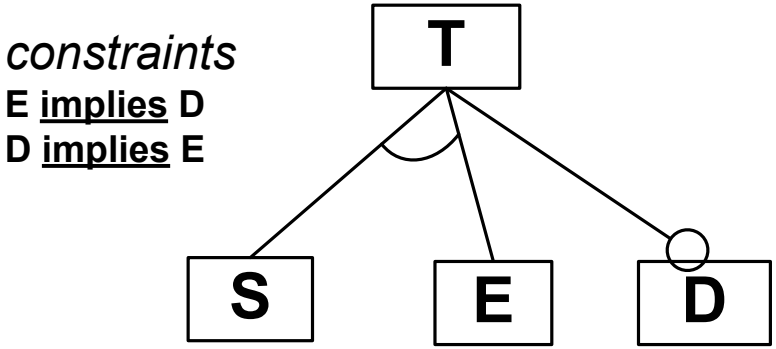
Hierarchy and Configuration matter!

Slicing Operator

slicing criterion : an arbitrary set of features, relevant for a feature model user

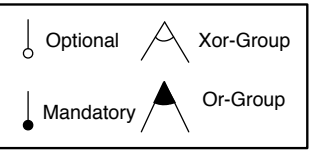


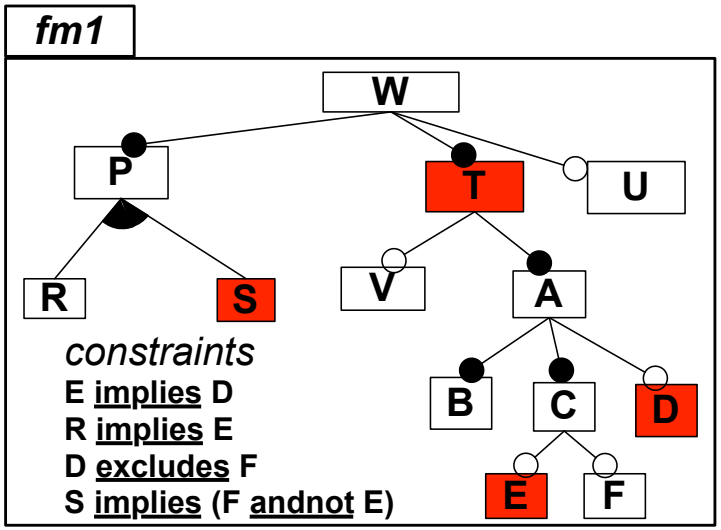
slice : a new feature model, representing a projected set of configurations



Slicing operator: going into details

projected set of configurations



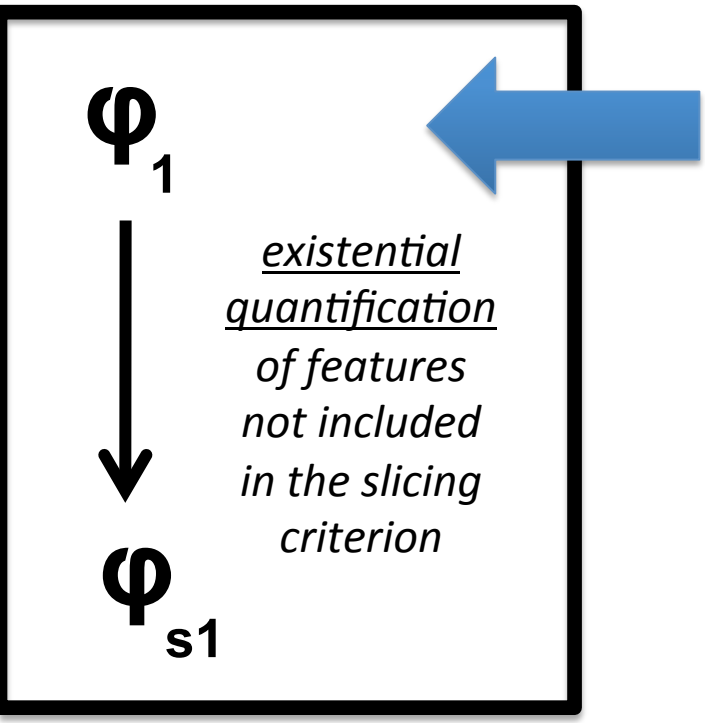


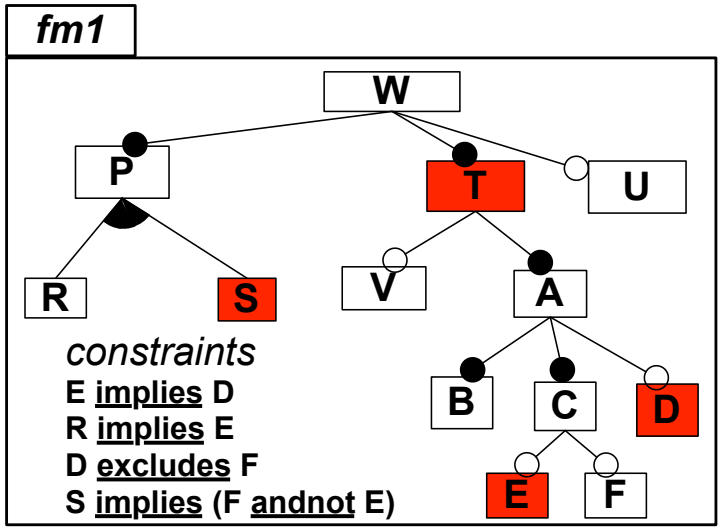
fm1 = {
{A,B,C,D,E,P,R,T,U,W},
{A,B,C,P,S,T,U,W},
{A,B,C,D,E,P,R,T,W},
{A,B,C,P,S,T,V,W},
{A,B,C,P,S,T,U,V,W},
{A,B,C,P,S,T,W},
{A,B,C,D,E,P,R,T,V,W},
} }

fm1p = {
{D,E,T},
fm1p = {
{S,T},
{D,E,T},
{D,E,T},
{S,T},
{S,T},
{S,T},
{S,T},
{D,E,T}
}

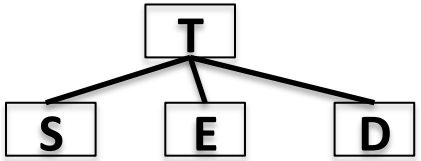
Slicing operator: going into details

synthesizing the corresponding feature model

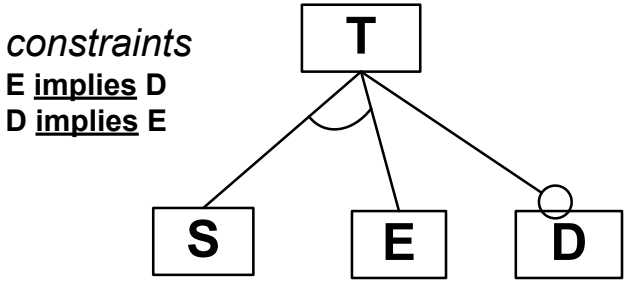




+



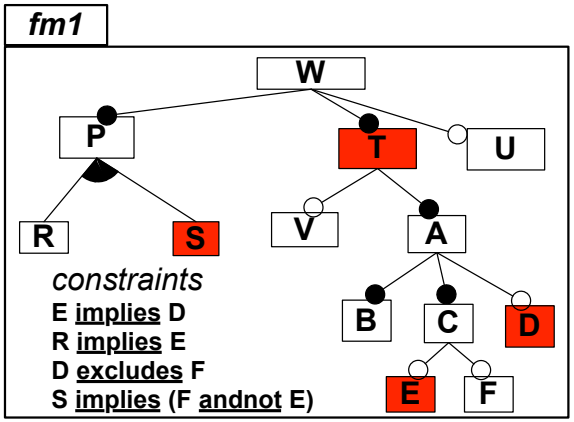
fm1p = {
{D,E,T},
{S,T}
}

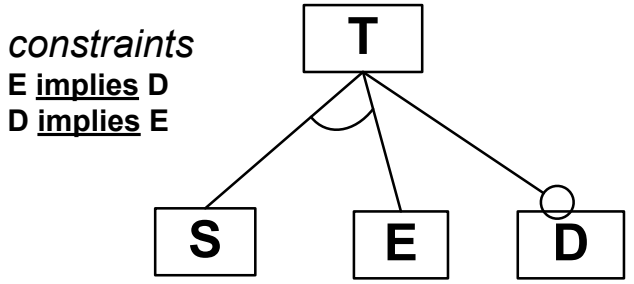


see also [Acher et al., ASE'11/AOSD'12]

Slicing operator with FAMILIAR (1)

```
fm1 = FM (W : P T [U] ; T : [V] A ;
                          A : B C [D] ;
                           C : [E] [F] ;
                           P : (R|S)+ ;
                           E implies D ; R implies E ;
                           S implies (F and !E) ; D implies !F ; )

fm2 = slice fm1 including { S T E D }
fm2bis = slice fm1 excluding { W P R V A B C F U }

cmp = compare fm2 fm2bis
assert (cmp eq REFACTORING)
```





slicingOp.fml

Slicing with FAMILIAR (2)

```
fm1 = FM (W : P T [U] ; T : [V] A ;
                          A : B C [D] ;
                           C : [E] [F] ;
                           P : (R|S)+ ;
                           E implies D ; R implies E ;

                           S implies (F and !E) ; D implies !F ; )

fm2 = slice fm1 including fm1.A.* ++ { fm1.A }

fm3 = slice fm1 including fm1.P.* ++ { fm1.P }
//fm3bis = slice fm1 including { fm1.P fm1.R fm1.S } // equivalent to fm3

fm4 = slice fm1 including { fm1.E fm1.D fm1.F }

fts5 = { fm1.P fm1.W } ++ fm1.P.*
fm5 = slice fm1 including fts5
```

Revisiting Merge and Slice: AggregateMerge

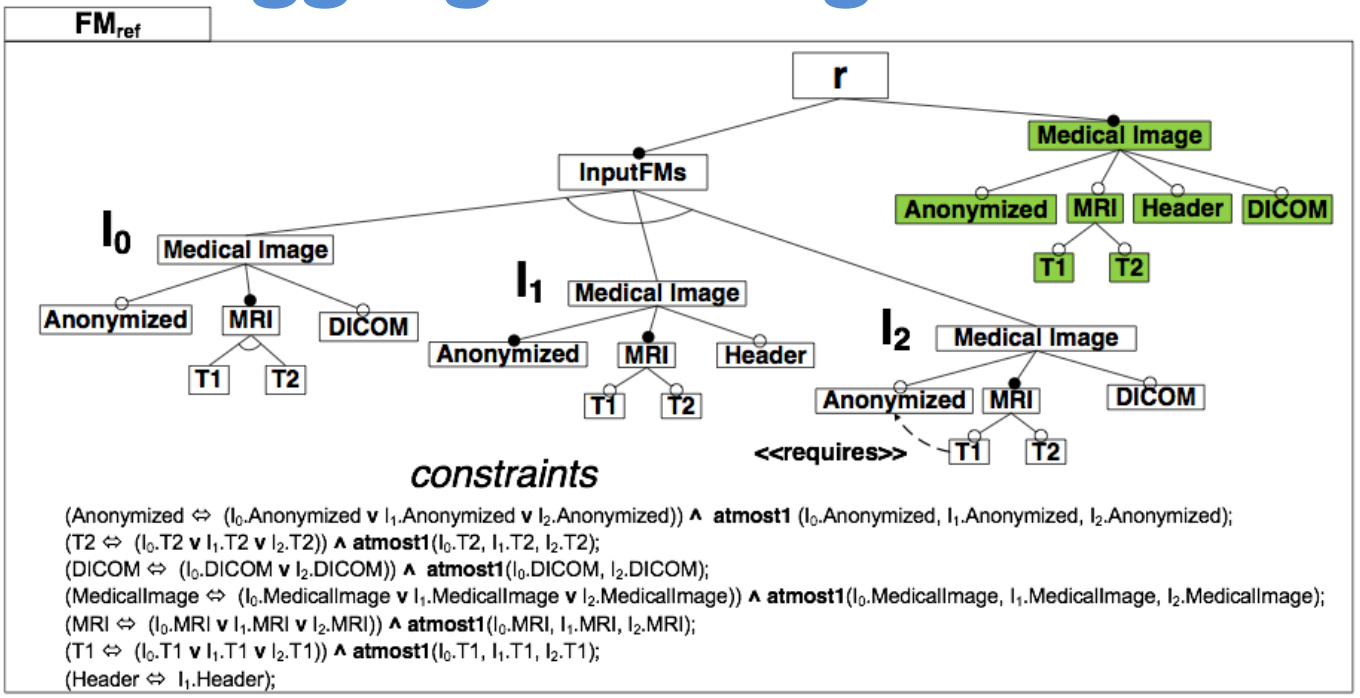
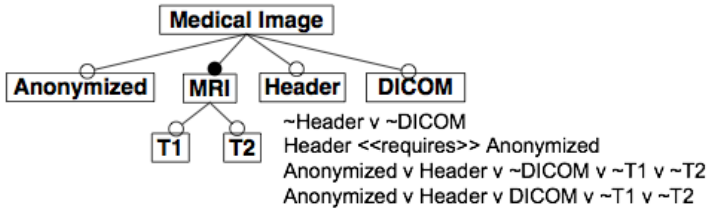


Figure 7.10: Merge of three feature models, I_0, I_1 and I_2 using the slicing operator

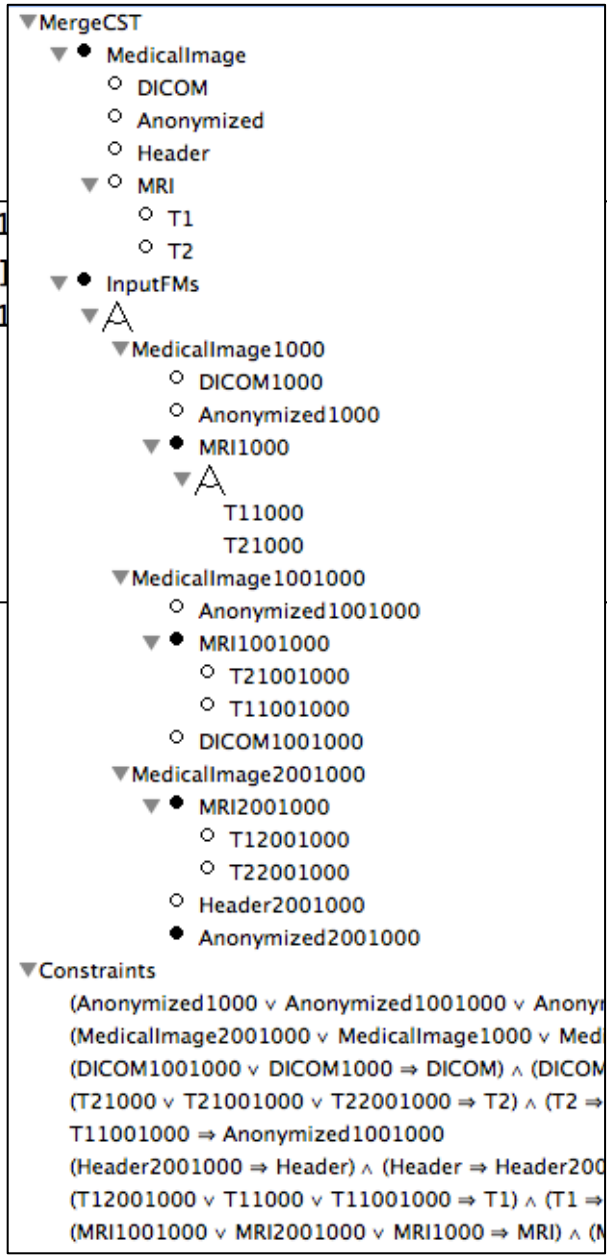


mergeWithAggregateMI.fml

Revisiting Merge and Slice: AggregateMerge

```
fmsupp1 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : (T1
fmsupp2 = FM (MedicalImage : Anonymized MRI [Header] ; MRI : [T1]
fmsupp3 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : [T1

fmSupp = merge sunion fmsupp*
fmSuppAgg = aggregateMerge sunion fmsupp*

fmSuppAggSlice = slice fmSuppAgg including fmSupp.*
```

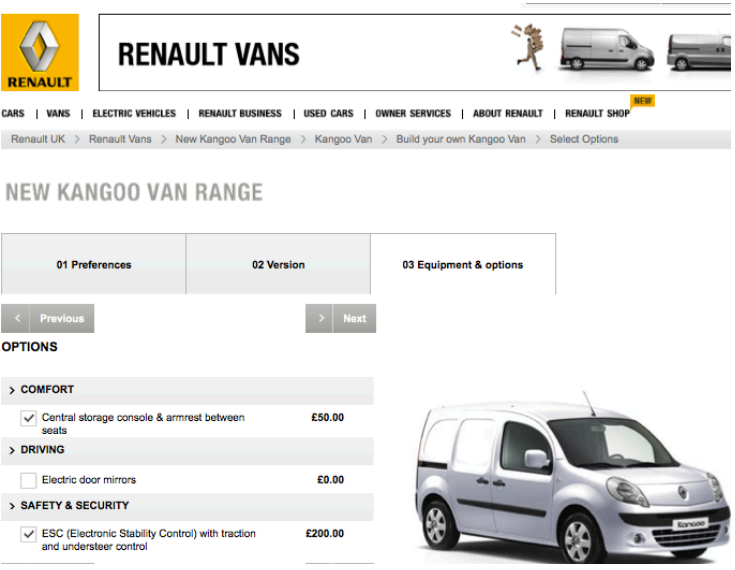
```
fml> compare fmSuppAggSlice fmSupp
res4: (STRING) REFACTORING
```

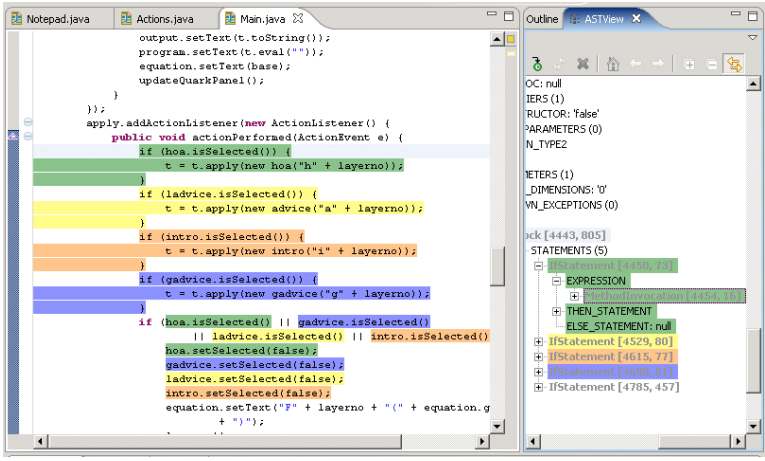


Putting all together: Example 2

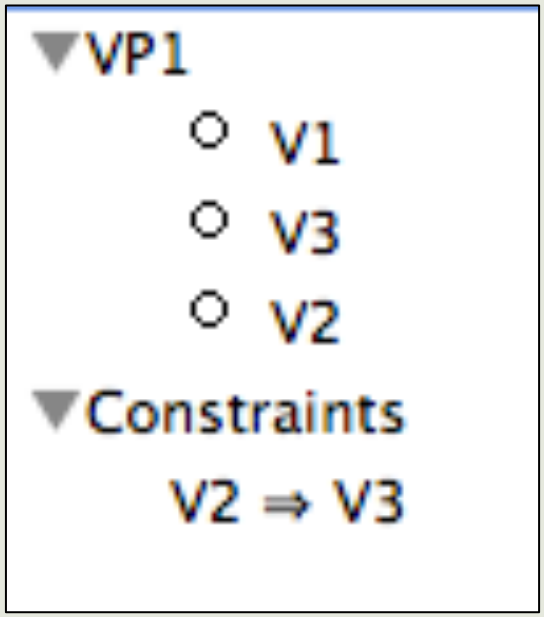
From marketing, customers, product management

From existing software assets (technical variability)





Metzger, Heymans et al. "Disambiguating the Documentation of Variability in Software Product Lines: A Separation of Concerns, Formalization and Automated Analysis" (RE'07)

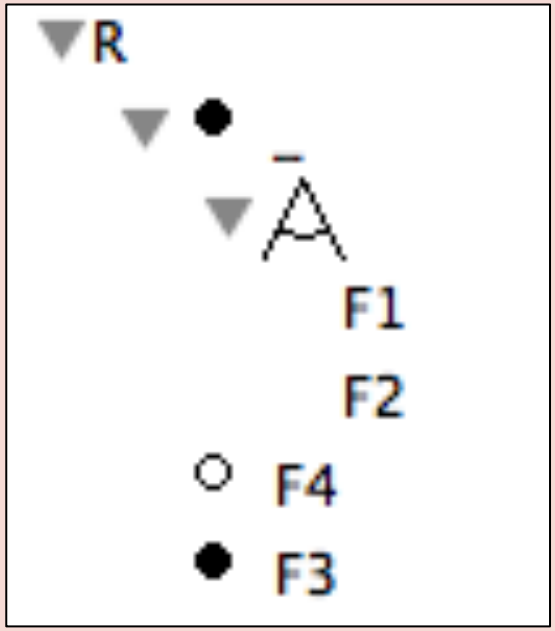


From marketing, customers, product management

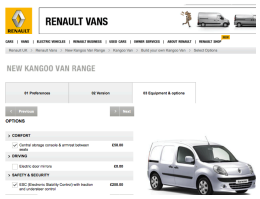
usefulness

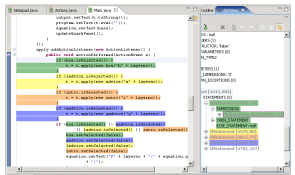
V1 ⇔ f1
V2 ⇔ f2
V3 ⇔ f3

realizability



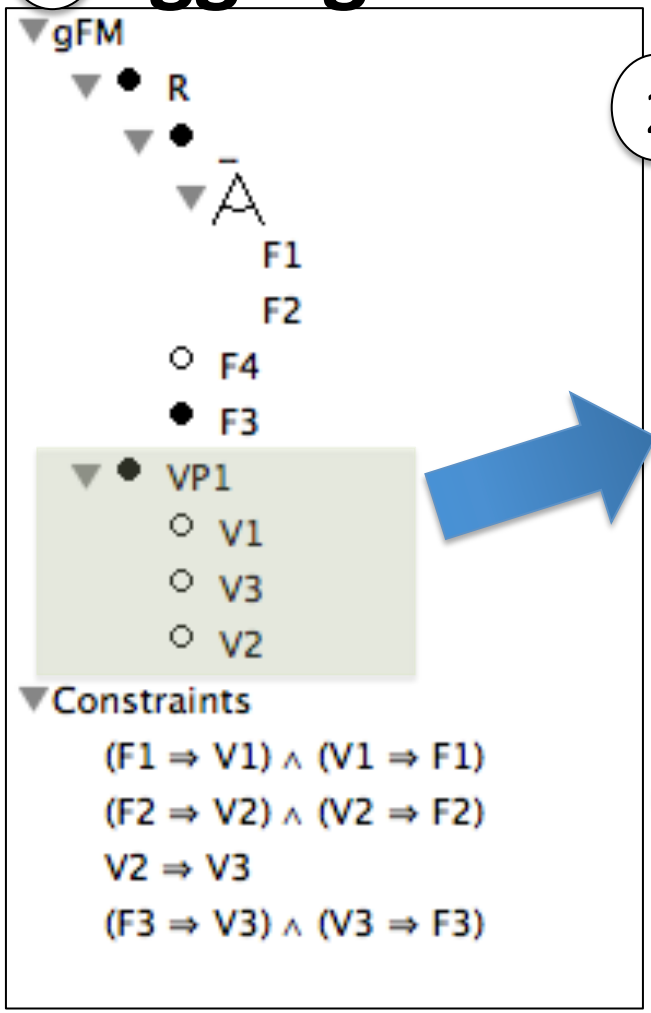
From existing software assets



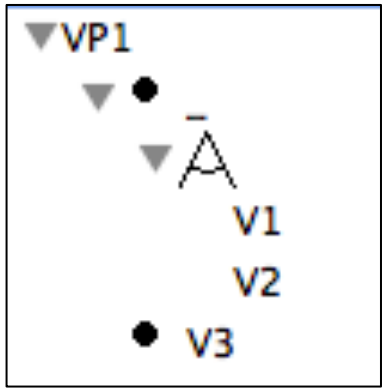


Realizability checking

1 **aggregate**



2 **slice** (“realizable part”)



3 **compare**

φ

{{V1,V3,V2,VP1},
{V1,VP1},
{V3,VP1},
{VP1}}

4 **merge diff**

(“unrealizable products”)

see also [Acher et al. AOSD’12 and CAiSE’12

realizibility.fml

With FAMILIAR

```
/*
* Metzger et al. 2007, RE'07
* Disambiguating the .....
* Figure 1, Section 3
*/
```

```
fmSoftware = FM (R : (F1|F2) F3 [F4] ; )
```

```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.6.jar realizability.fml
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)  version 0.9.9.6
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> ls
(FEATURE_MODEL) gFM
(FEATURE_MODEL) fmSoftware
(FEATURE_MODEL) fmPLDiff
(FEATURE_MODEL) fmPLPrime
(FEATURE_MODEL) fmPL
(SET) xLink
fml> configs fmPLDiff
res1: (SET) {{VP1;V1};{VP1;V3};{VP1};{V3;V1;VP1;V2}}
```

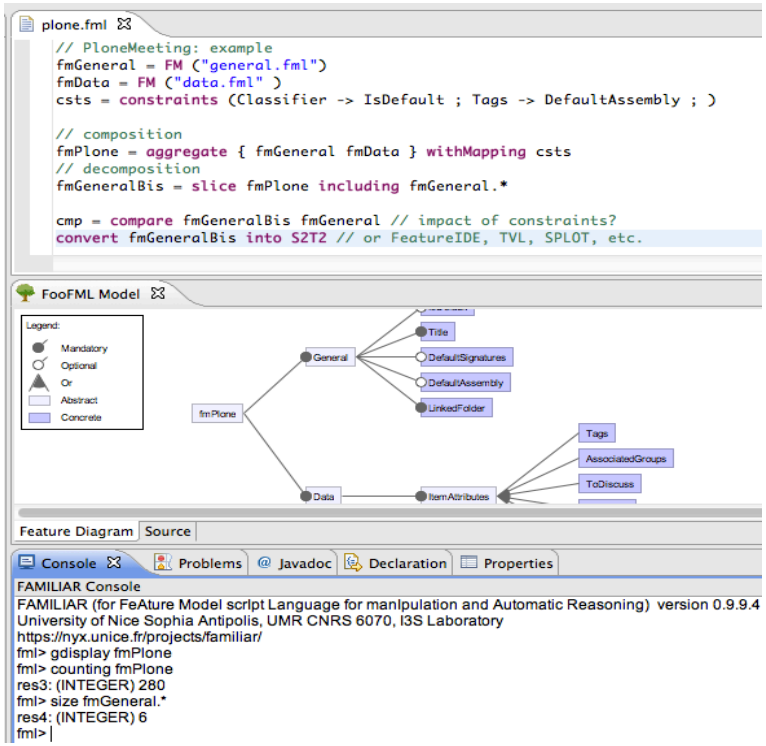
3. Model-based Variability Management:

Summary

What's next?

Summary

```
root PloneMeeting {
    group allof {
        General,
        Data,
        WorklowSec,
        Interface,
        Email,
        Tasks,
        Advices,
        Votes
    }
}

General {
    group allof {
        Title ,
        opt DefaultAssembly,
        opt DefaultSignatures,
        LinkedFolder,
        opt IsDefault,
        NumberLastItemLastMeeting {
            int number;
        },
        opt NumberLastMeetingConfig {
            int number;
        },
        opt MeetingConfigID
    }
}
```



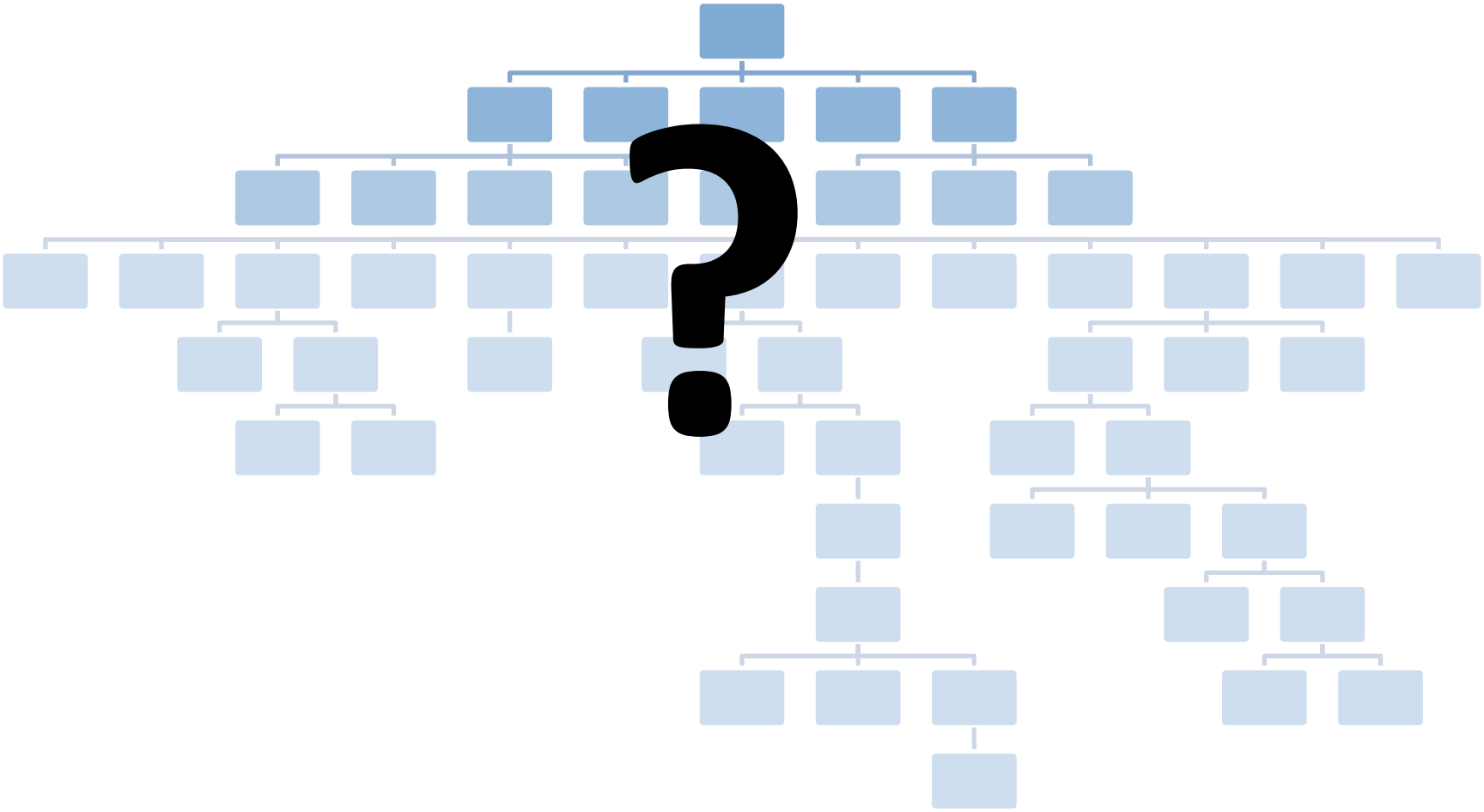
Text-based variability language with formal semantics & support (Boolean form)

A domain-specific language for Importing, exporting, editing, composing, decomposing, computing differences, comparing, reasoning about (multiple) feature models

What's next?

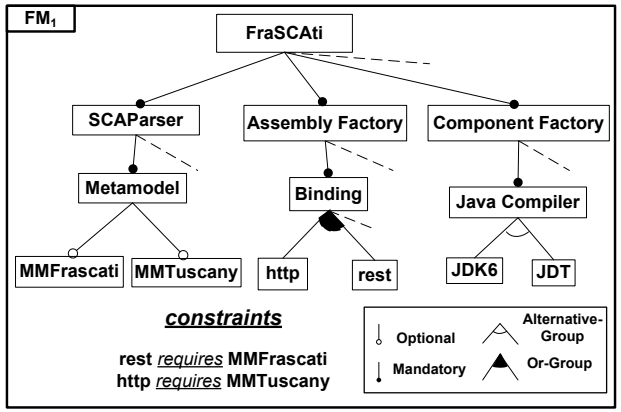
- Increasing further the adoption of the languages
  - You're the future users!
- TVL
  - Comprehensive support for non Boolean constructs
  - Satisfiability Modulo Theory (SMT) solver
- FAMILIAR
  - Applicability, Learnability, Expressiveness, Usability

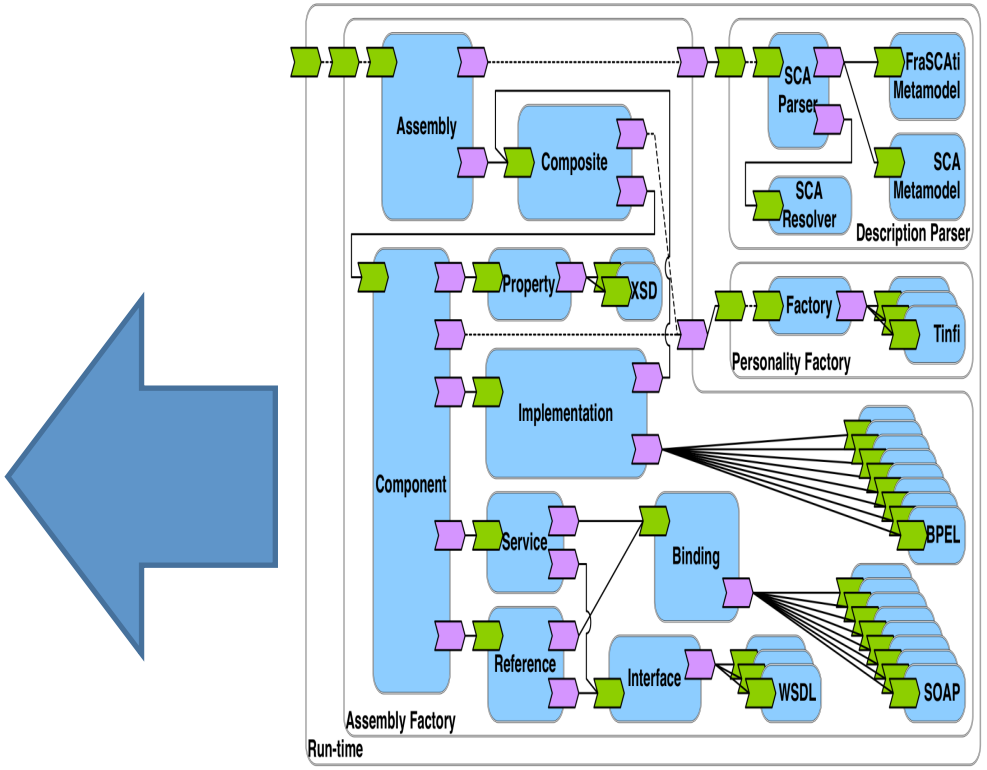
- Connection of feature models to other artefacts
  - Automated product derivation
  - Verification & Certification



#1 Reverse Engineering Architectural Feature Models

Case Study: FraSCAti Architecture

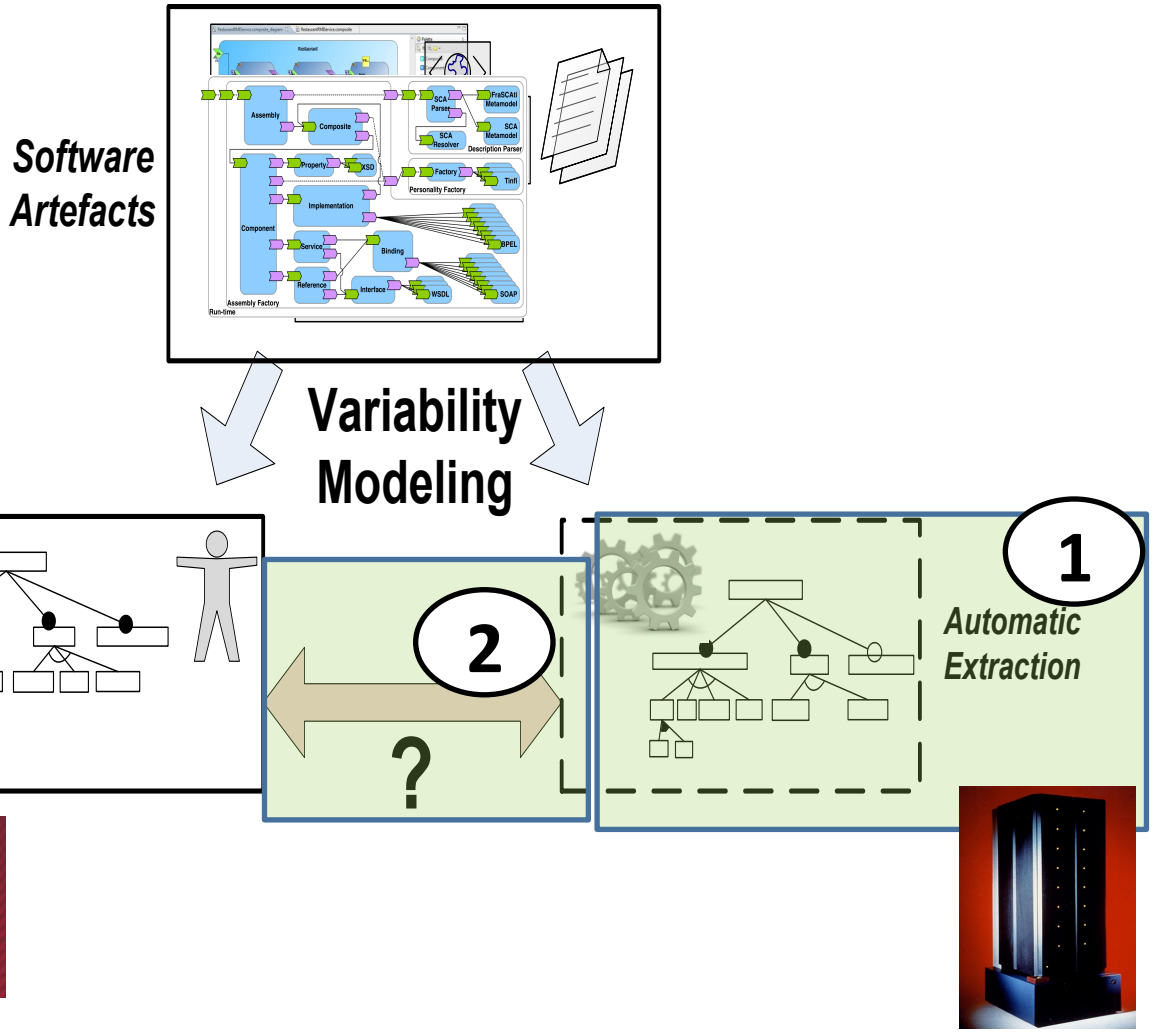




[Acher et al., ECSA'11]
[Acher et al., BENEVOL'11]
[Acher et al., GDR GPL'12]

Collaboration with Anthony Cleve (University of Namur / PRECISE, Belgium), Philippe Collet and Philippe Lahire (University of Nice Sophia Antipolis), Philippe Merle and Laurence Duchien (University of Lille / INRIA)

Extraction process



Philippe Merle,
software architect of FraSCAti

Combination of plugin dependencies and hierarchical component model to synthesise a feature model

Highlights

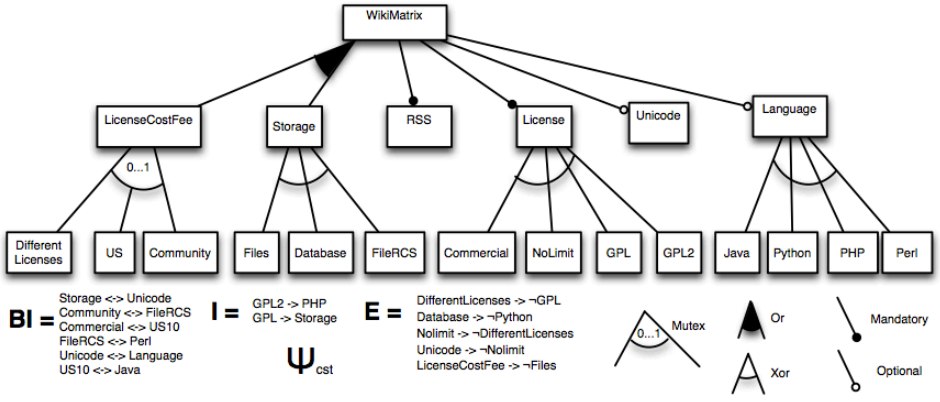
- Automated Procedure
  - Extracting and Combining Variability Sources

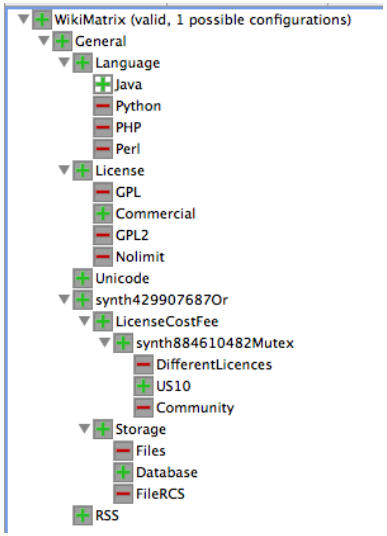
  (incl. software architect knowledge)
  - Advanced feature modeling techniques have been developed (tool supported with FAMILIAR)
- Lessons Learned
  - Extraction procedure yields promising results
  - Essential role of software architect
    - To validate the extracted feature model
    - To integrate knowledge
- Ongoing Work
  - Evolution of FraSCAti (v1.3, v1.4, etc.)
  - Applicability to other software architectures

Identifier	License	Language	Storage	LicenseCostFee
Confluence	Commercial	Java	Database	US10
PBwiki	Nolimit	No	No	Yes
MoinMoin	GPL	Python	Files	No
DokuWiki	GPL2	PHP	Files	No
PmWiki	GPL2	PHP	Files	No
DrupalWiki	GPL2	PHP	Database	Different Licences
TWiki	GPL	Perl	FilesRCS	Community
MediaWiki	GPL	PHP	Database	No


```
<features>
  <product p_id="1" name="DokuWiki">
    <general_info name="General Information">
      <info name="Description">
        DokuWiki is a standards compliant, simple to use Wiki, mainly aimed at creating documentation of any kind. It is ta
        syntax which makes sure the datafiles remain readable outside the Wiki and eases the creation of structured texts. Al
      </info>
      <info name="Record added">2005-11-22</info>
      <info name="Last updated">2011-05-25</info>
    </general_info>
    <itemgroup g_id="1" name="General Features">
      <item i_id="1" name="Version">2011-05-25 "Rincewind"</item>
      <item i_id="60" name="Last Release Date">2011-05-25</item>
      <item i_id="2" name="Author">Andreas Gohr</item>
      <item i_id="3" name="URL">http://www.dokuwiki.org</item>
      <item i_id="4" name="Free and Open Source">Yes</item>
      <item i_id="5" name="License">GPL 2</item>
      <item i_id="51" name="Programming Language">PHP</item>
      <item i_id="123" name="Data Storage">Files</item>
      <item i_id="55" name="License Cost/ Fee">No</item>
      <item i_id="61" name="Development status">Mature</item>
      <item i_id="118" name="Intended Audience">private, small to medium companies</item>
    </itemgroup>
    <itemgroup g_id="18" name="Hosting Features">
      <item i_id="140" name="Storage Quota">Linux, UNIX, Windows, MacOS X, probably others</item>
      <item i_id="141" name="Bandwidth Quota">No</item>
      <item i_id="142" name="Other Limits">Apache, IIS, Lighttpd, anything with PHP support</item>
      <item i_id="146" name="Topic Restrictions"/>
      <item i_id="143" name="Corporate Branding">Yes</item>
      <item i_id="144" name="Own Domain">No</item>
```


#2 from product descriptions to feature models



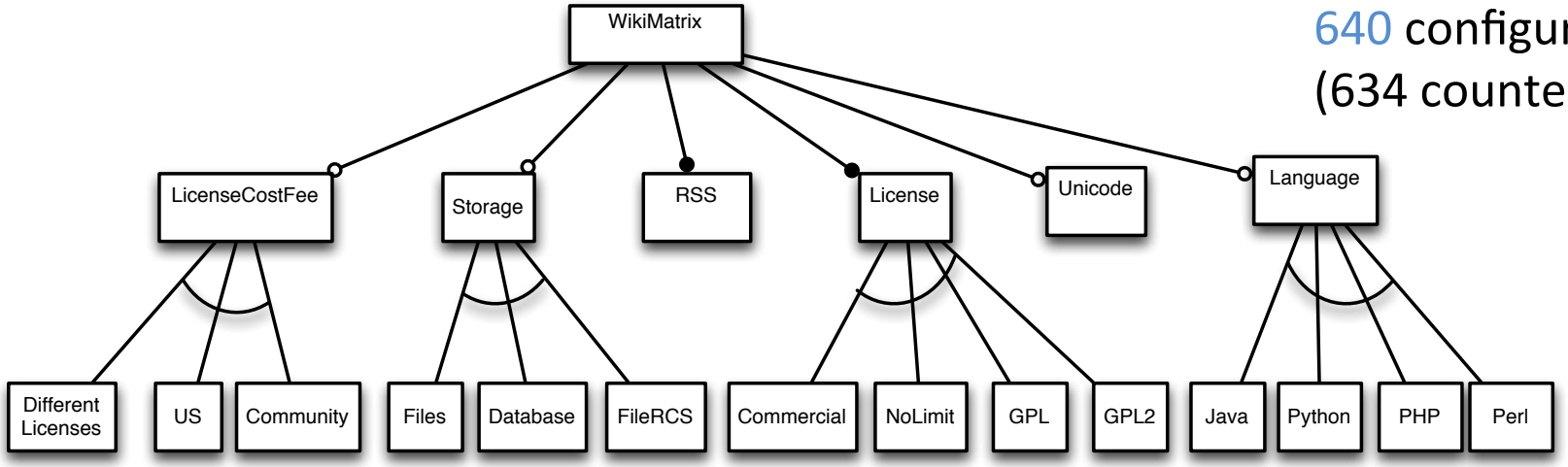


[Acher et al., VaMoS'12]

Collaboration with Patrick Heymans, Anthony Cleve, Gilles Perrouin (University of Namur / PRECISE, Belgium), Philippe Collet and Philippe Lahire (University of Nice Sophia Antipolis),

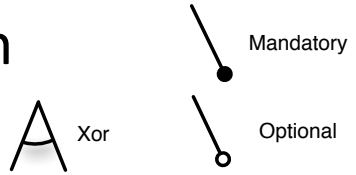
Manual extraction of a feature model from product description(s) is not possible

Identifier	License	Language	Storage	LicenseCostFee	RSS	Unicode
Confluence	Commercial	Java	Database	US10	Yes	Yes
PBwiki	Nolimit	No	No	Yes	Yes	No
MoinMoin	GPL	Python	Files	No	Yes	Yes
DokuWiki	GPL2	PHP	Files	No	Yes	Yes
PmWiki	GPL2	PHP	Files	No	Yes	Yes
DrupalWiki	GPL2	PHP	Database	Different Licences	Yes	Yes
TWiki	GPL	Perl	FilesRCS	Community	Yes	Yes
MediaWiki	GPL	PHP	Database	No	Yes	Yes



640 configurations
(634 counter examples)

Exact set of configurations, each configuration corresponding to at least one product

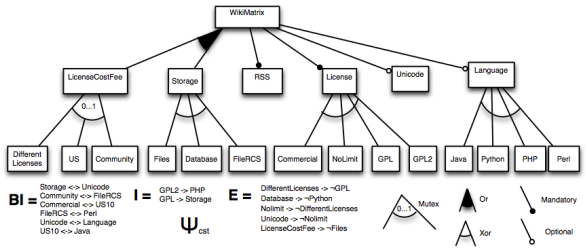


Automation

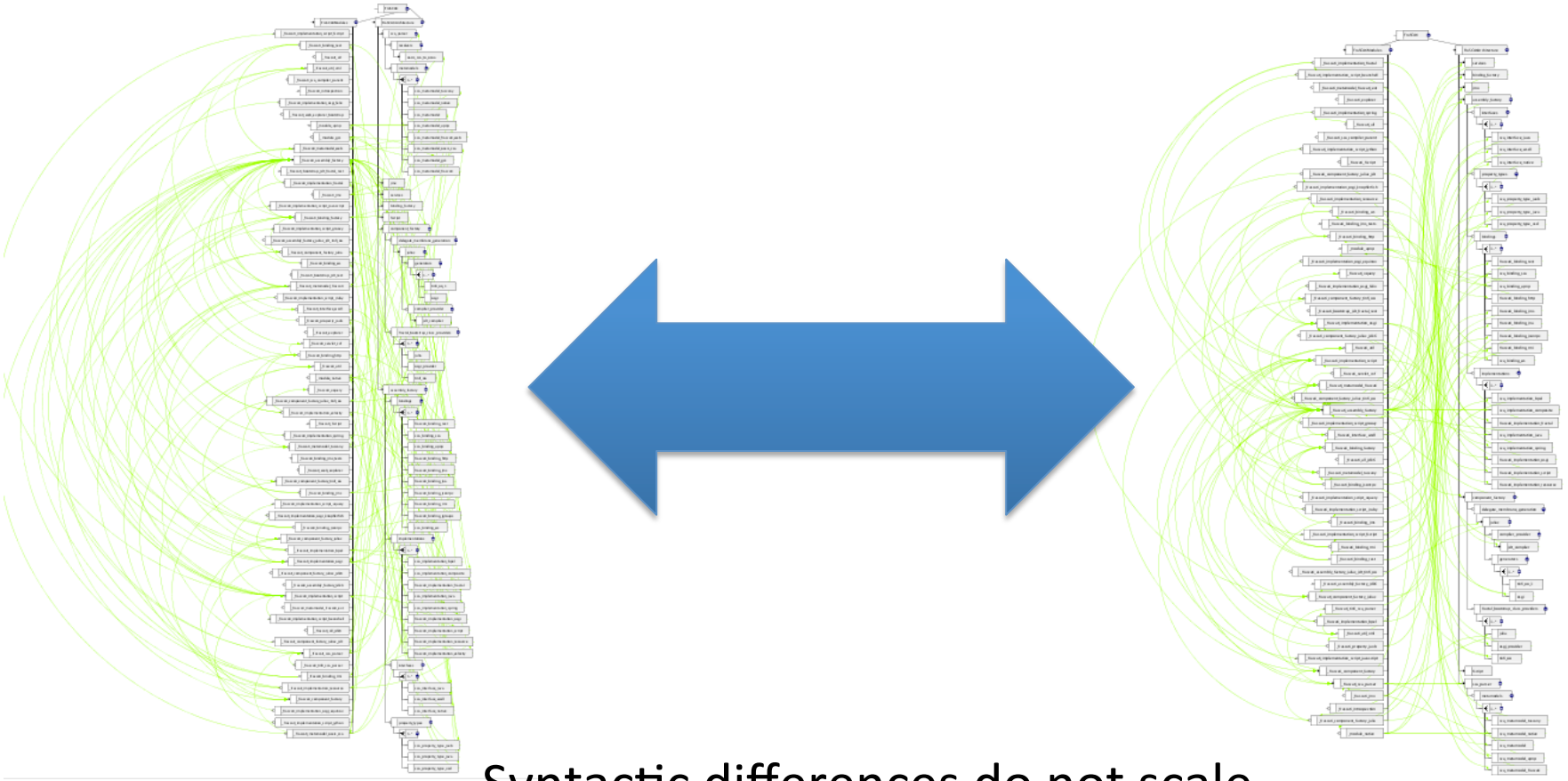
- Each product description is encoded as a feature model

Identifier	License	Language	Storage	LicenseCostFee	RSS	Unicode	
Confluence	Commercial	Java	Database	US10	Yes	Yes	→ fm1
PBwiki	Nolimit	No	No	Yes	Yes	No	→ fm2
MoinMoin	GPL	Python	Files	No	Yes	Yes	→ fm3
DokuWiki	GPL2	PHP	Files	No	Yes	Yes	→ fm4
PmWiki	GPL2	PHP	Files	No	Yes	Yes	→ fm5
DrupalWiki	GPL2	PHP	Database	Different Licences	Yes	Yes	→ fm6
TWiki	GPL	Perl	FilesRCS	Community	Yes	Yes	→ fm7
MediaWiki	GPL	PHP	Database	No	Yes	Yes	→ fm8

- Feature models {fm1, fm2,…,fm8} are **merged**
  - Output: a new feature model
    - Configuration: union of input sets of configurations
    - Hierarchy: by default, we exploit the structure of the tabular data
      - Can be overridden by specific user directive
  - VariCell
    - DSL built on top of FAMILIAR



Feature Model Differences

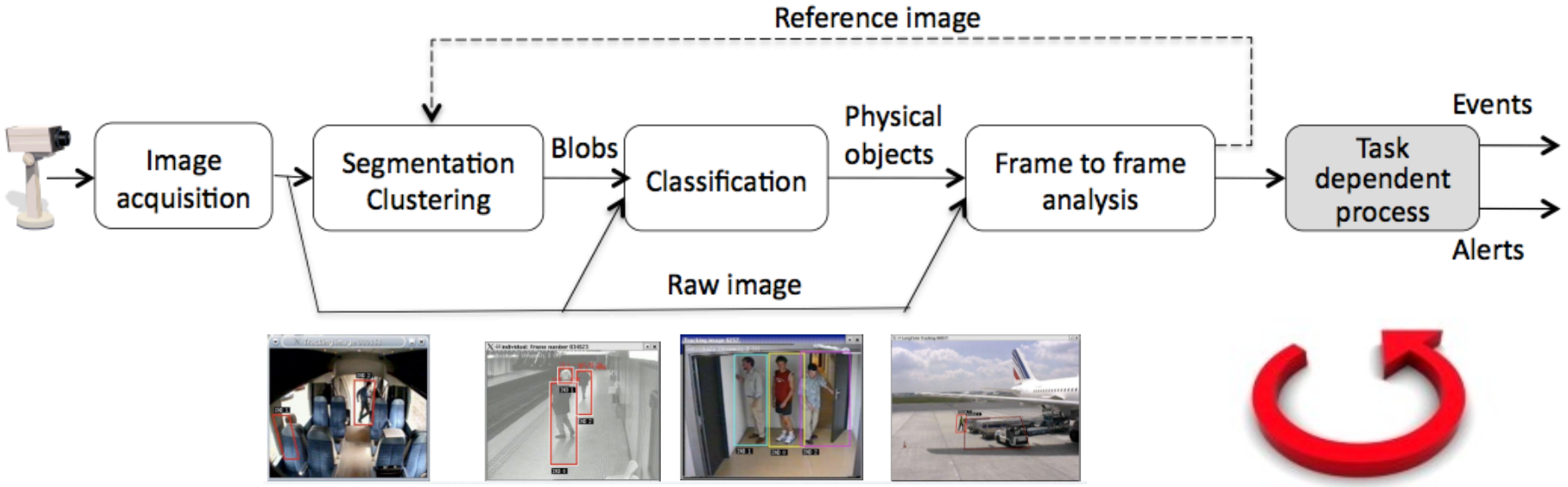


Syntactic differences do not scale

see [Acher et al., CAiSE'12]

Modeling Variability From Requirements to Runtime

The case of video surveillance processing chains



large number of software configurations

for a large number of requirements

Adaptive systems

Collaboration with Sabine Moisan and Jean-Paul Rigault (INRIA)

Contributions

- Separation of Concerns
  - Software variability is distinguished from requirements variability
- A systematic approach to specify and reason about variability of adaptive systems
  - Variability requirements are step-wised specified at design time
    - Some variability choices are kept for runtime adaptation (e.g., Day/Night)
  - Reduction of software configurations to be considered at runtime

- Reasoning operations
  - Consistency checking
  - "Reachability" property
    - for all valid requirements (e.g., contexts), there exists at least one valid software configuration
  - Specialization checking
  - Choices Propagation
- Language support
  - aggregate + slice + compare + editing facilities
  - Reuse of feature models and analysis procedures in 6 scenarios

See also [Acher et al., ICECCS'11] or Demonstration (screencast)