Software Product Lines and Variability Management

Philippe Collet (Professor, Univ. Nice Sophia Antipolis)

With some slides from Mathieu Acher

Course Agenda

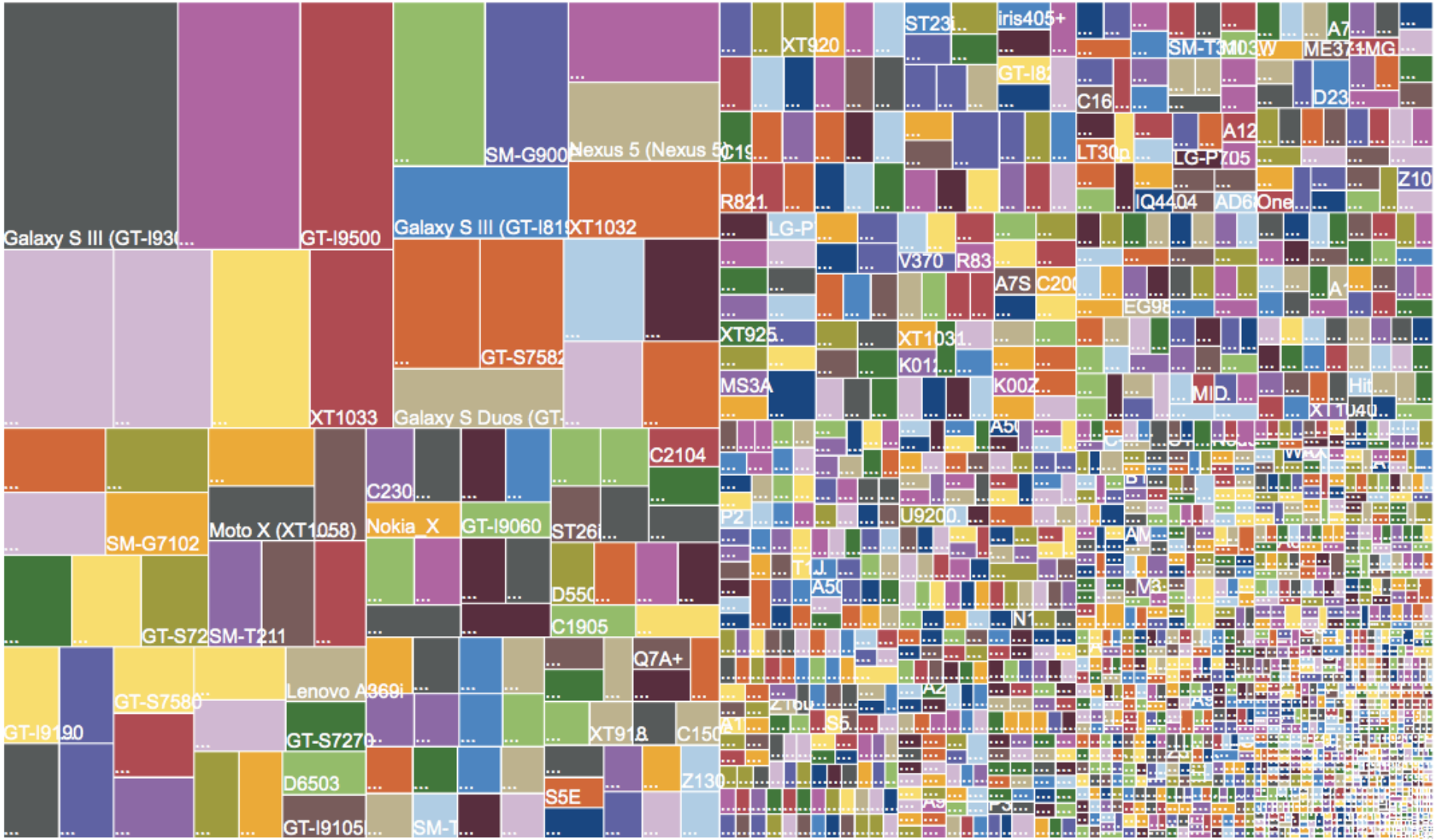
- Why managing **Variability** does matter
- **Managing** variability models with Feature Models
- **FAMILIAR** : a DSL for large scale management of Feature Models
- Some **Applications** of Feature Modeling

Why managing Variability does (and will) matter

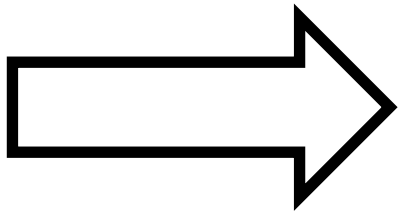
http://opensignal.com/reports/2014/android-fragmentation/

Android, survey on 682,000 devices

- August 2013: 11,868 distinct devices
- August 2014: 18,796 distinct devices



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

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

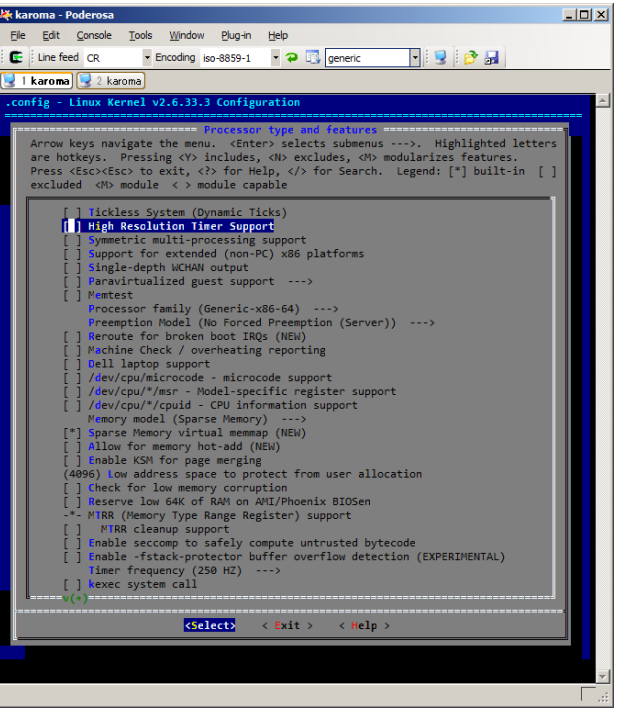
Legend

- ☑ 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

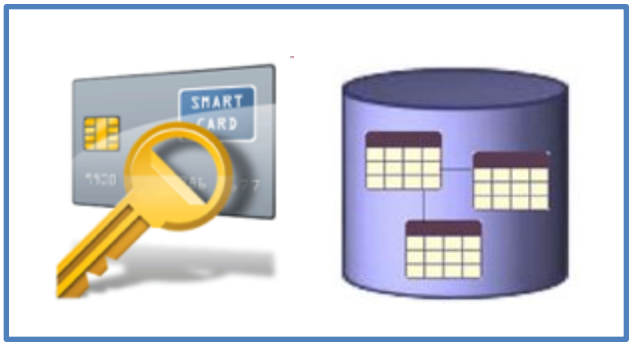
Pricing Details

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

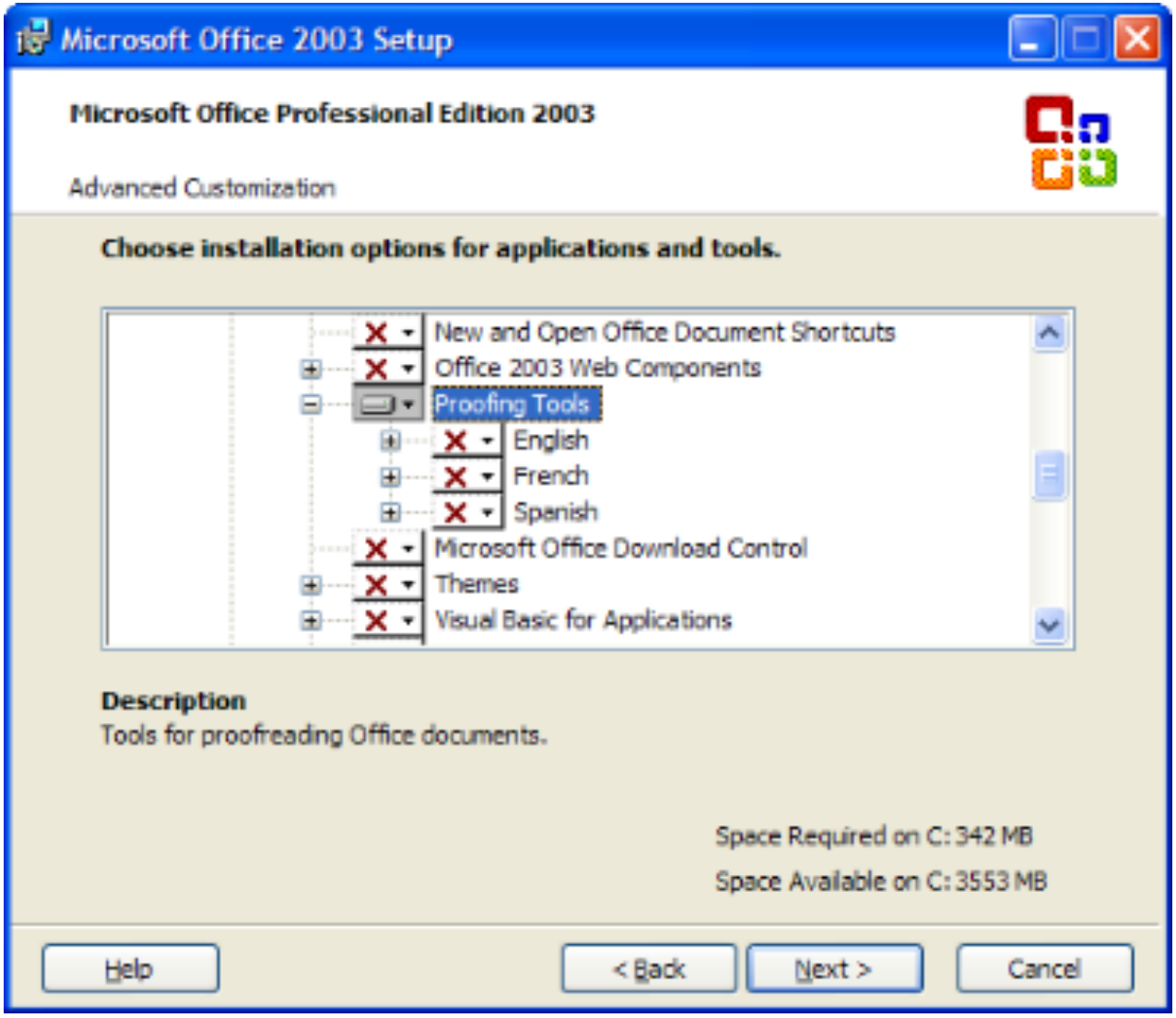


Database Engine



Printer Firmware

Features in Microsoft Office



Hall of Fame

splc.net/fame.html

BOSCH
Invented for life

hp
invent

GM

LSI LOGIC

PHILIPS

ERICSSON

SALiON
TARGET. WIN. DELIVER.

BOEING

MARKET MAKER AG

NOKIA
Connecting People

Lucent Technologies
Bell Labs Innovations

Cummins

CelsiusTech

Software Product Lines
Family of Systems
Configurable Systems
Feature-rich Systems
Family of Systems
Dynamically Adaptive Systems

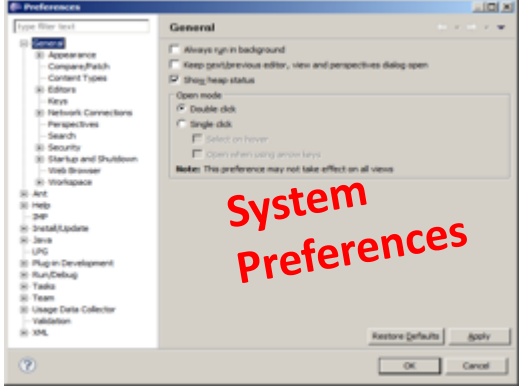
Variability

« A set of programs is considered to constitute a **family**, whenever it is worthwhile to study programs from the set by **first studying the common properties** of the set and then determining the **special properties** of the individual family members »

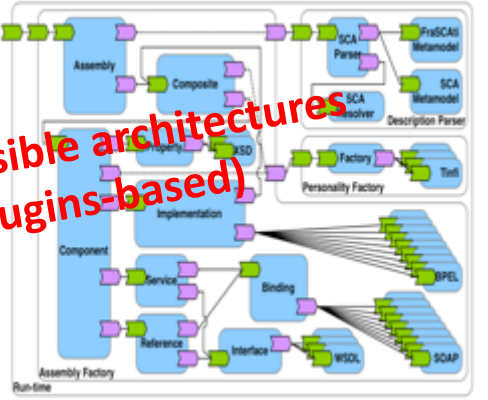
aka **Variability**

David L. Parnas — "On the design and development of program families" in Transactions on Software Engineering, SE-2(1):1–9, 1976

Configurators



System Preferences



Extensible architectures (eg plugins-based)

Comparison of *

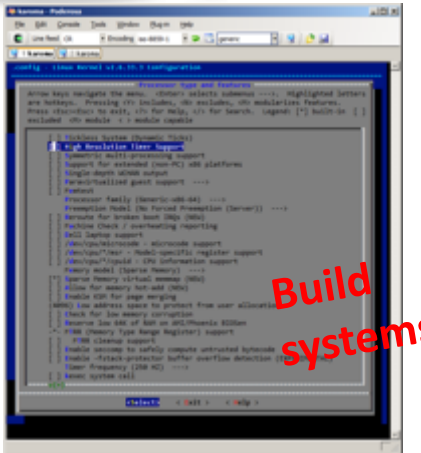
External Variability
Internal Variability
Variability @ run.time

Structural or behavorial models

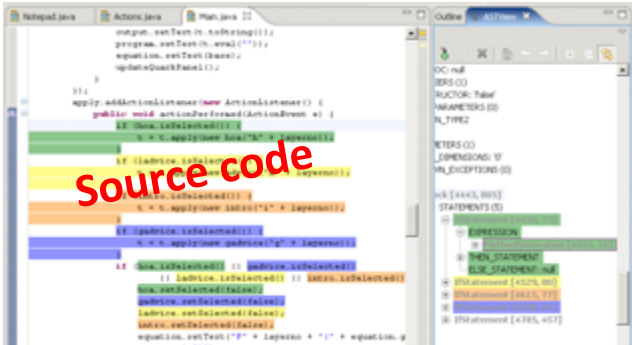
httpd.conf -- win32 Apache
Building a Web Server, for Windows

```
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>
```

Configuration files



Build systems



Source code

If you're able to master variability…

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

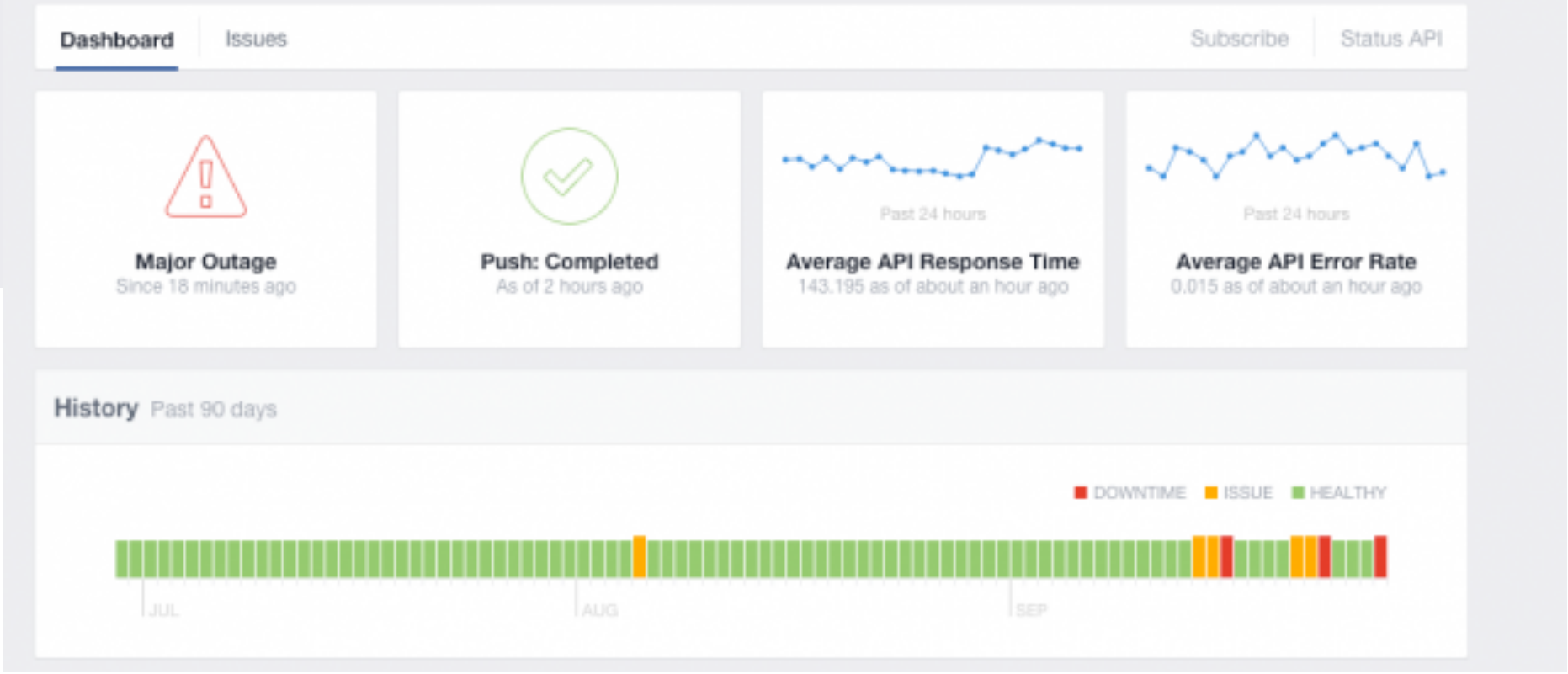
- But, are you able?
  - developing, verifying, certifying billions of variants is challenging!


Variability = Complexity
Christian Kästner slide

3 arrêts en septembre 2015 !

Facebook official message:

"We're currently restoring Facebook services that people had trouble accessing earlier today due to a configuration issue. We're working to bring things back to normal for everyone. We apologize to those who have been inconvenienced."



33 optional, independent features

a unique variant for every

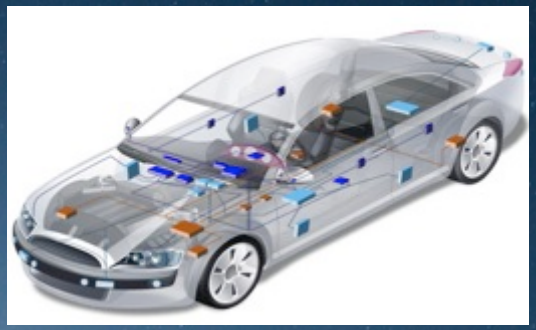
person on this planet

Christian Kästner slide

320 optional, independent features

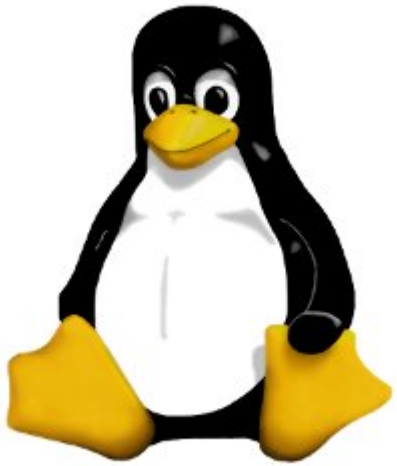
more variants than estimated

atoms in the universe



2000 features

10000 features



Christian Kästner slide


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

Unused flexibility


Illegal variant

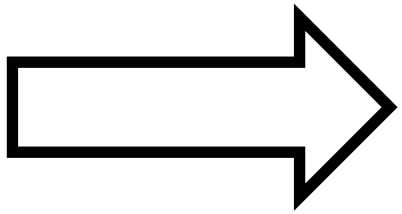
Why managing Variability does (and will) matter

Goal: **Software mass customization / Adaptive and configurable systems**

Problem: **Variability = Complexity**

Approach: **Model-based variability management**

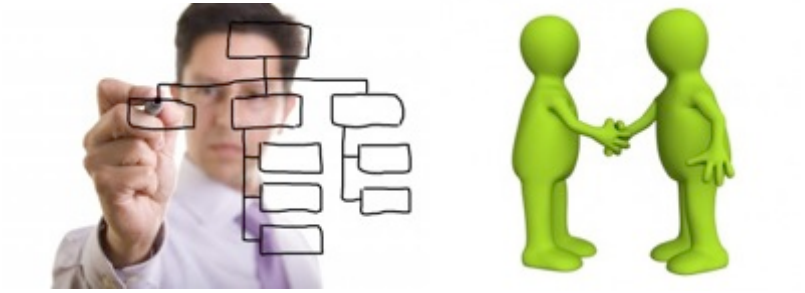
Software-intensive systems come in many variants



Model-based Variability Management

Modeling Variability

Communicative



Analytic

Generative

Software Product Lines: defintion

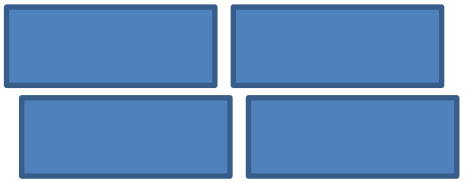
Product Lines Paradigm

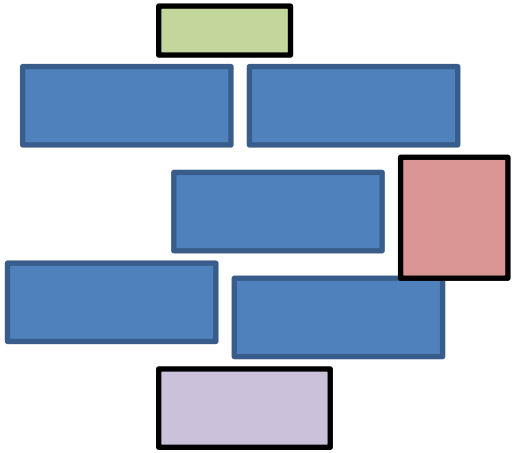
Systems of systems: Ultra Large Scale

a family of related software variants (= products) that share common properties and meet the requirements for a given domain.

Factoring out **commonalities**

for **Reuse** [Krueger et al., 1992] [Jacobson et al., 1997]

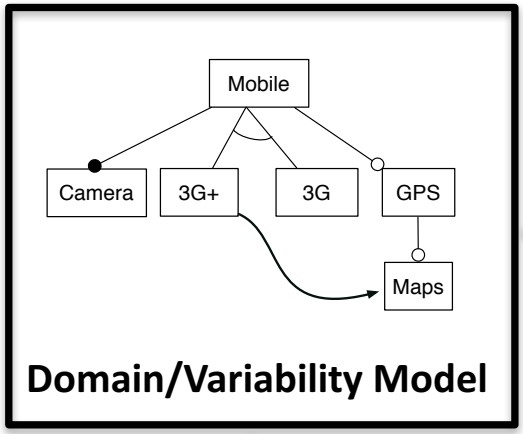


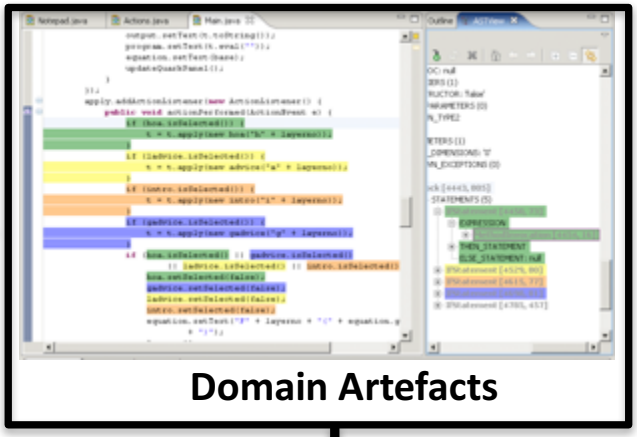


Managing **variabilities**

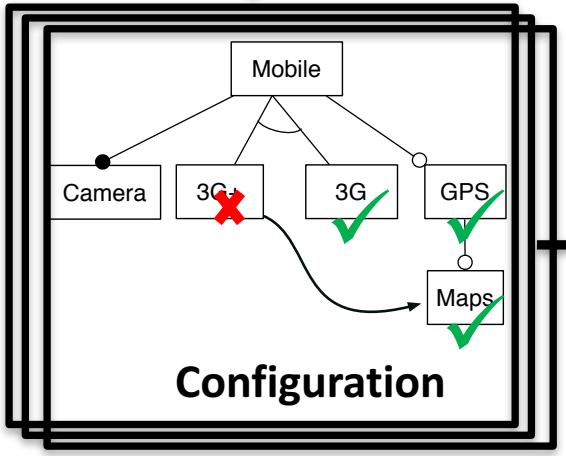
for Software **Mass Customization** [Bass et al., 1998] [Krueger et al., 2001], [Pohl et al., 2005]

Domain Engineering



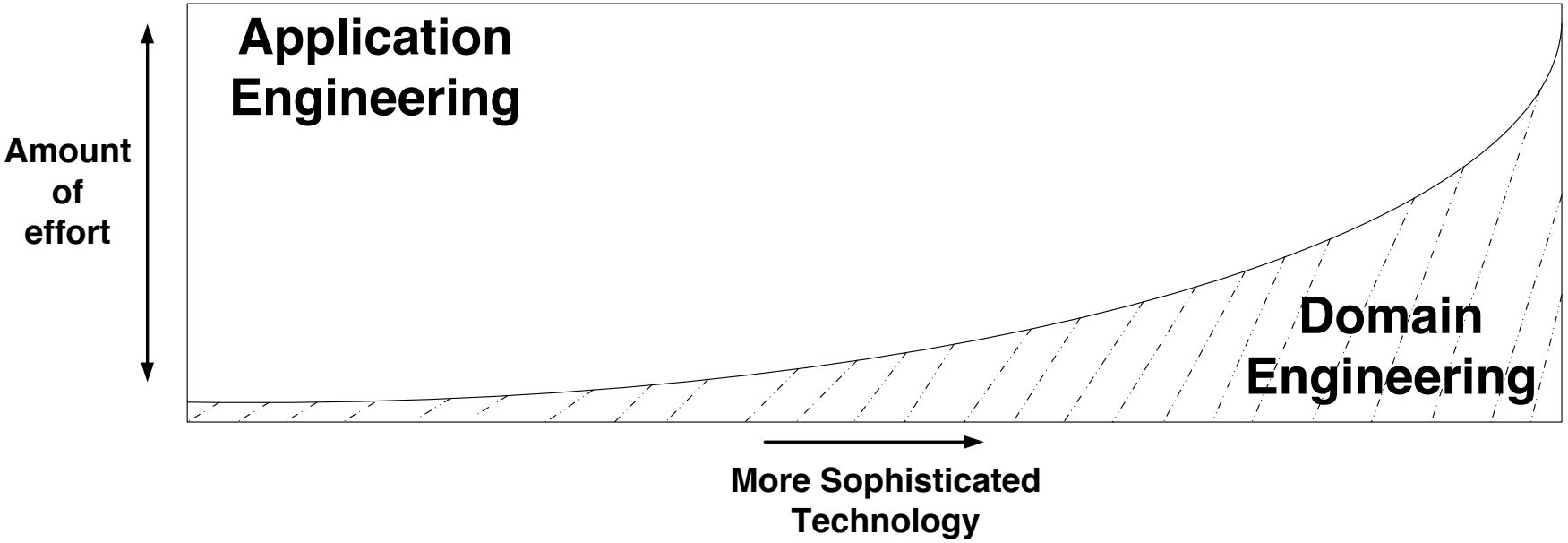


Application Engineering



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

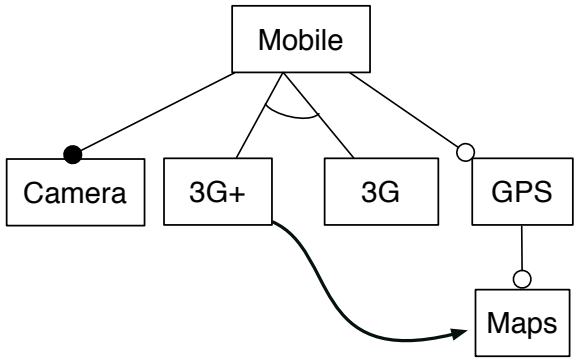
Jan Bosch et al. (2004)



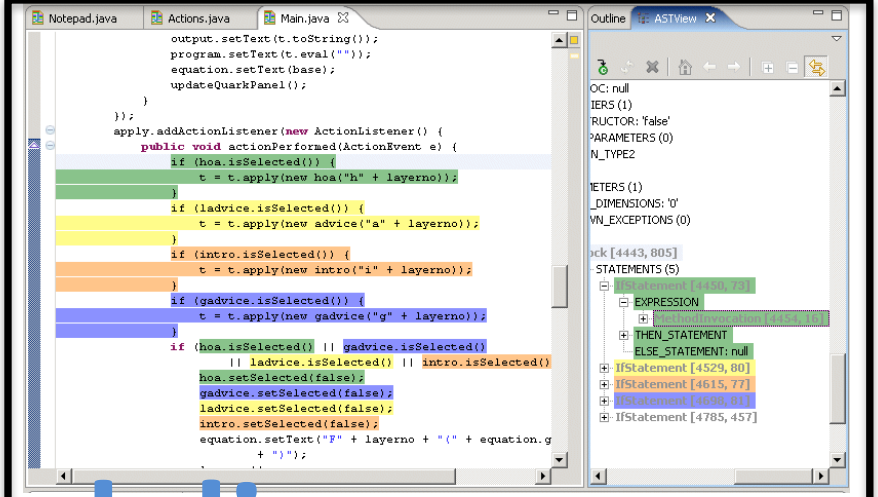
99% domain engineering,

1% application engineering?

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

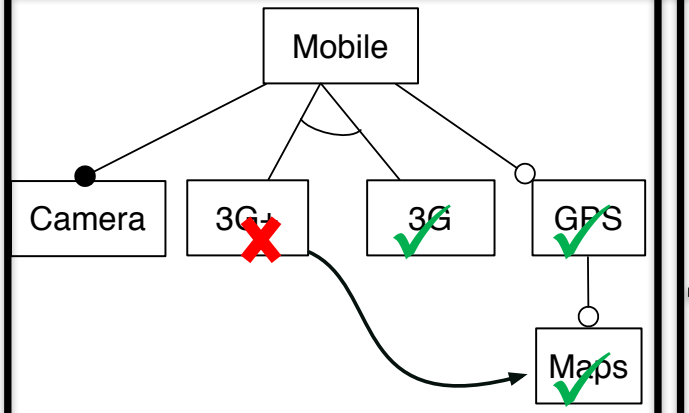


Variability Model



Domain Artefacts (e.g., source code)

Modeling variability is crucial



Configuration

Software Generator

Unused flexibility

Illegal variant

More Definitions

- **Domain** : Technology, business or knowledge sector that is characterized by a set of concepts and terminologies understandable by users of this sector
- **Variability** is formed by all assumptions showing how products, members of the SPL, differ among each other
- **Commonality** is formed by all assumptions that are true for all product members of the SPL

What is needed to handle Variability?

- Define **mandatory** properties and functionalities
- Define **optional** choices: 0, 1..N, choice among n
- **Variants**
- **Constraints**
  - Dependency
  - Mutual exclusion

How to express variability?

- Inheritance (design, implementation)
- Genericity (design, implementation)
- Design Pattern (design, implementation)
- Aspect-Oriented Design & Programming (design, implementation)
- Model Transformation (design)
- Model Composition (design)
- **Feature Models** (requirement engineering, design, also implementation)
  - Similar technique : **Decision Modeling**

Variability Model

Feature Model: *de facto* 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

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

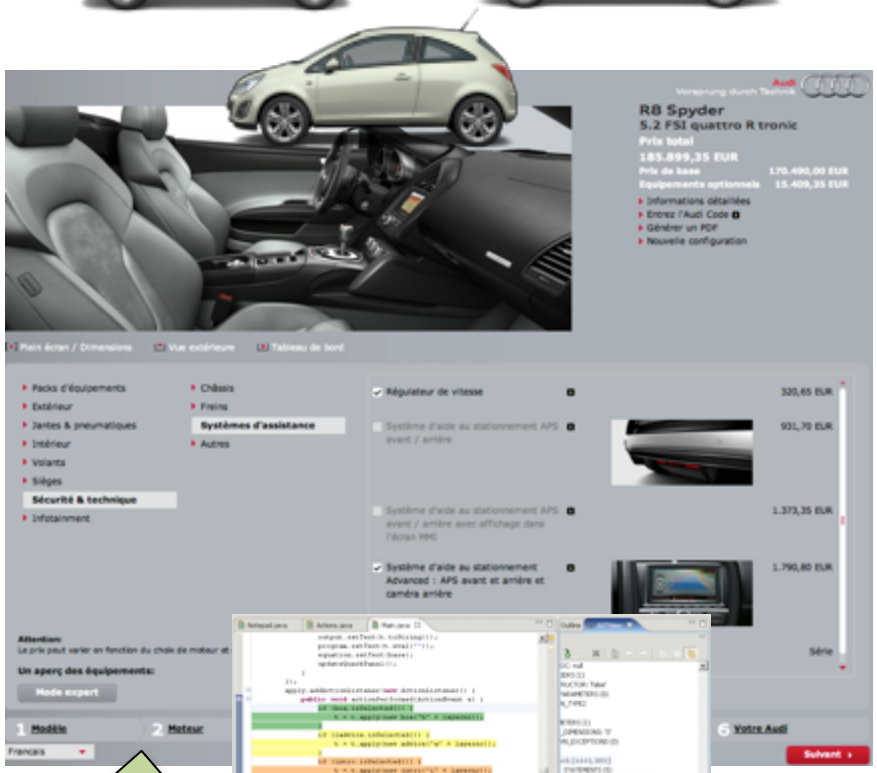
Réinitialiser la sélection

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

Francais

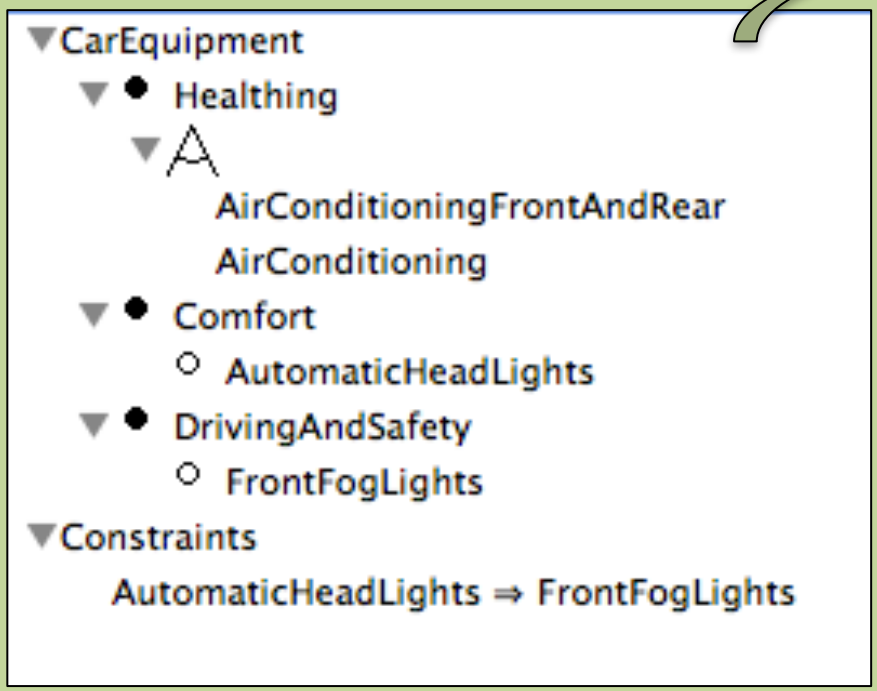
Suivant

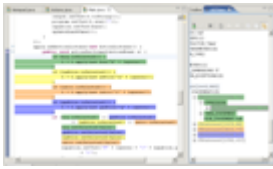
Feature Models



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

Feature Models (Background)

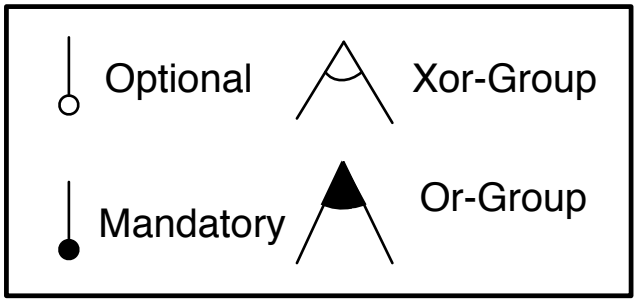


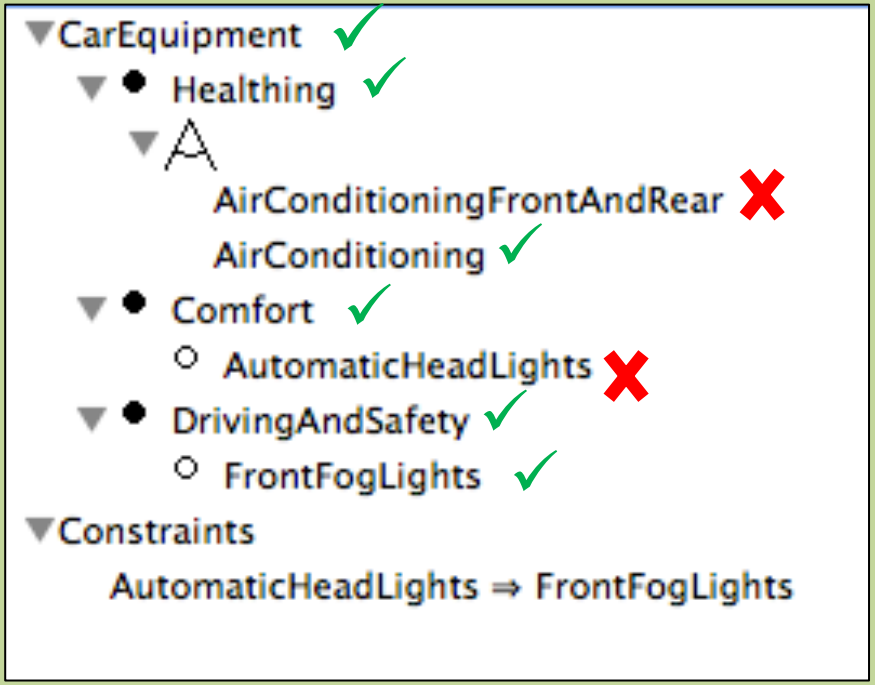


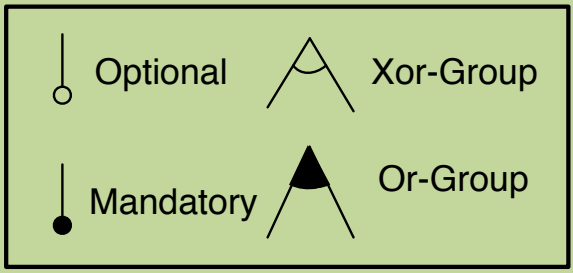
Hierarchy: rooted tree

Variability:

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



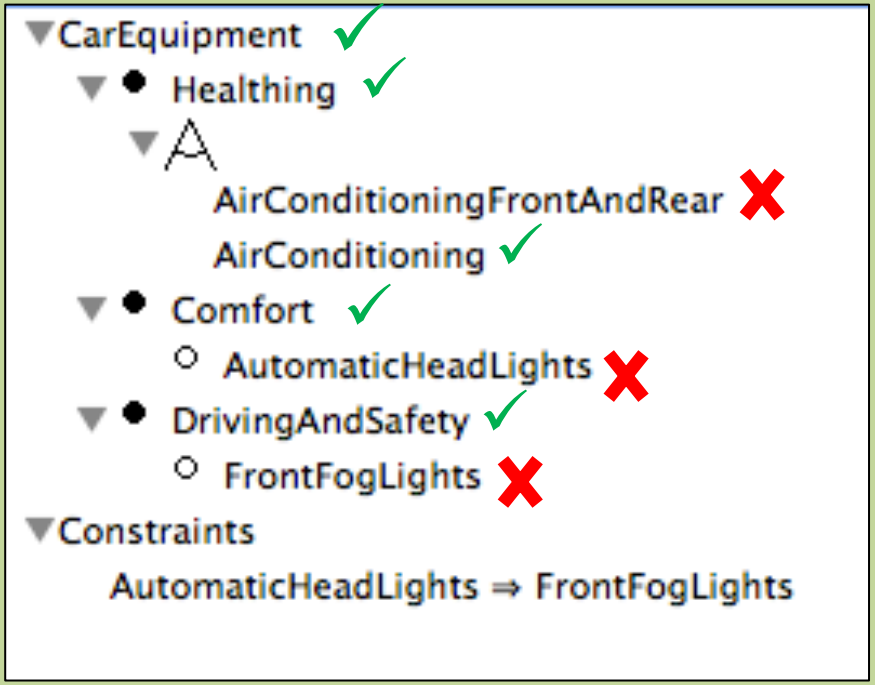


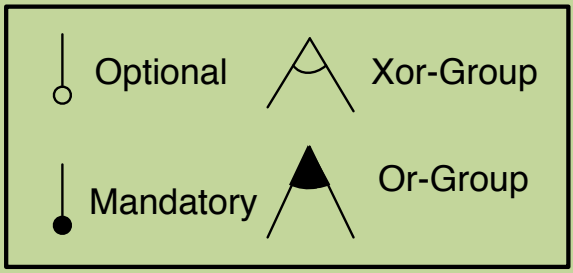


Hierarchy + Variability
=
set of valid configurations

configuration = set of features selected

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

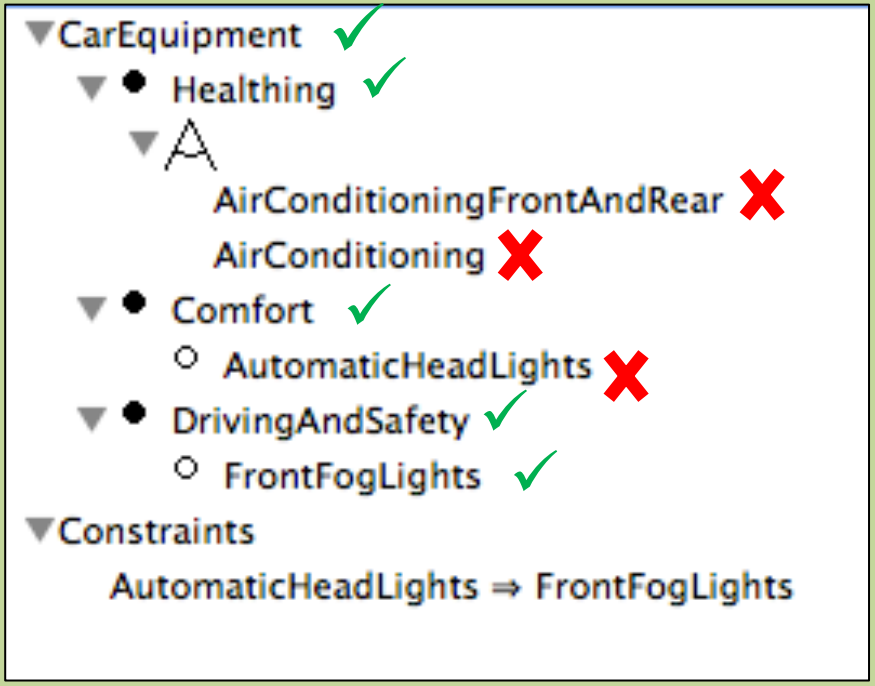


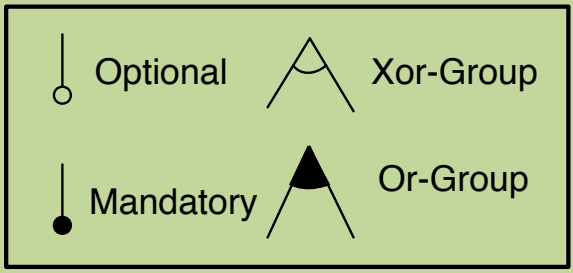


Hierarchy + Variability
=
set of valid configurations

configuration = set of features selected

{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning}

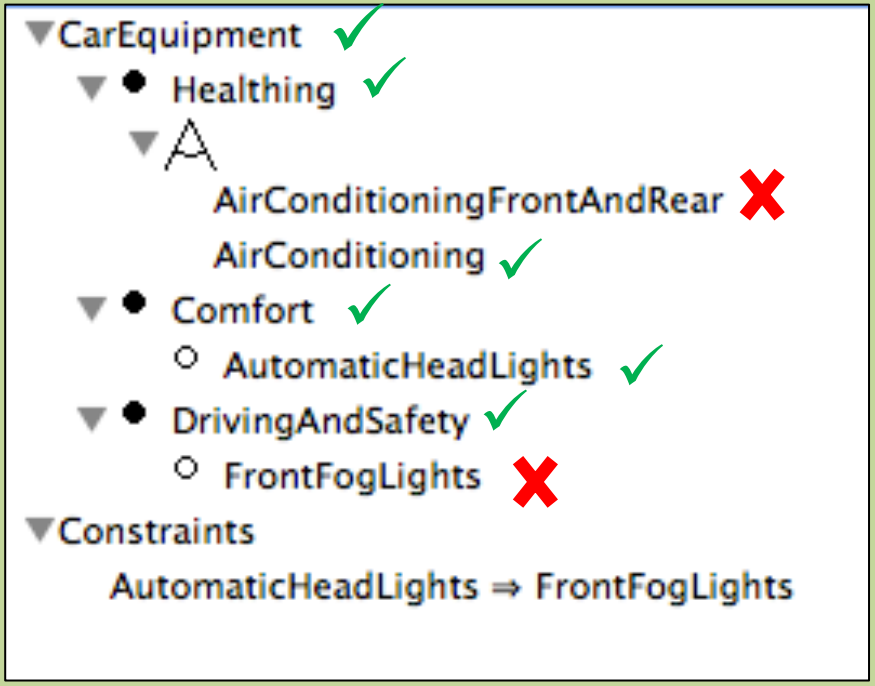


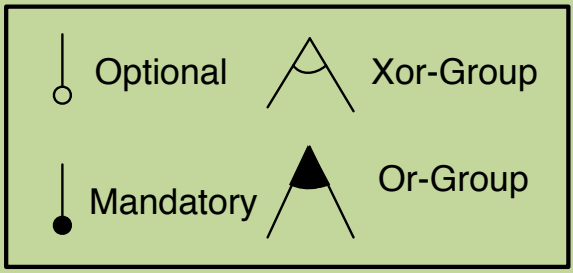


Hierarchy + Variability
=
set of valid configurations

configuration = set of features selected

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

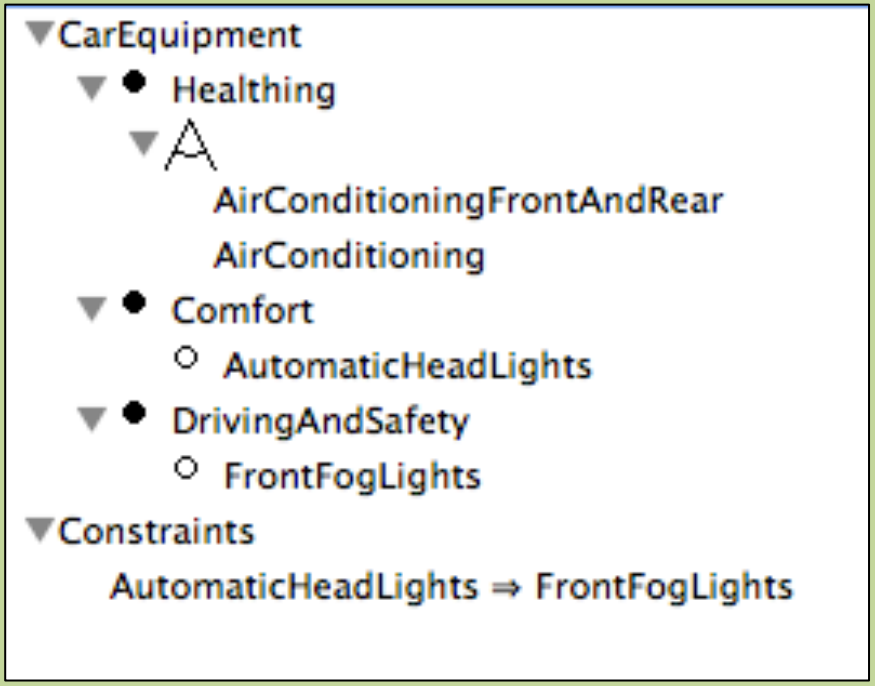


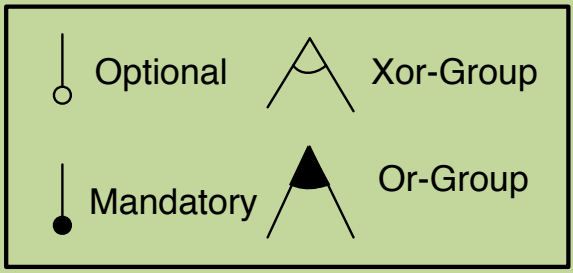


Hierarchy + Variability
=
set of valid configurations

configuration = set of features selected

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





Hierarchy + Variability
=
set of valid configurations

{CarEquipment, Comfort, DrivingAndSafety, Healthing}

X

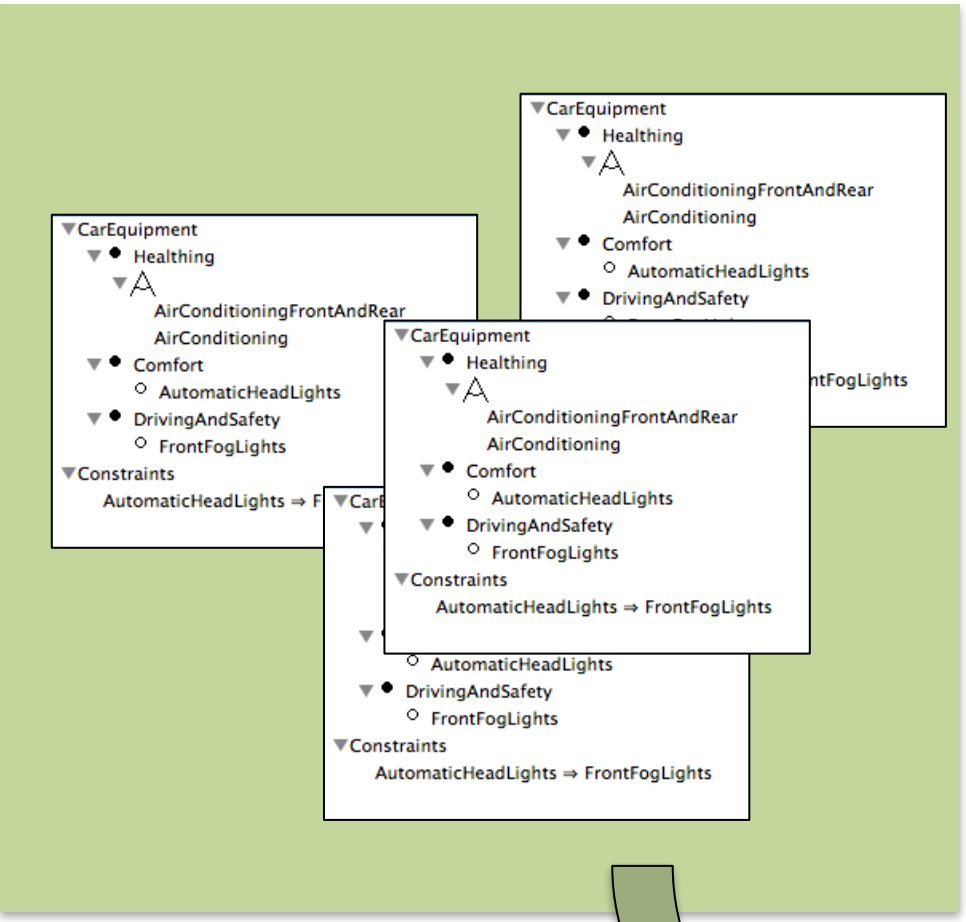
{AirConditioning, FrontFogLights}
{AutomaticHeadLights, AirConditioning, FrontFogLights}
{AutomaticHeadLights, FrontFogLights, AirConditioningFrontAndRear}
{AirConditioningFrontAndRear}
{AirConditioning}
{AirConditioningFrontAndRear, FrontFogLights}

Managing variability models with FAMILIAR

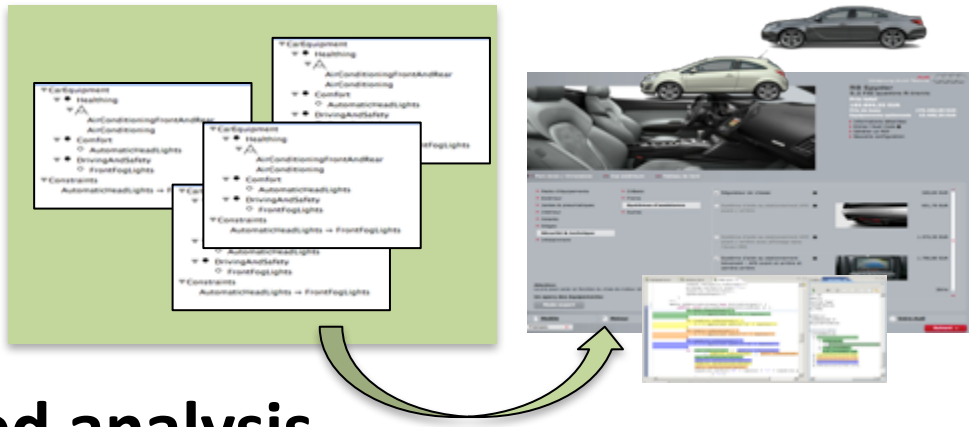
#1 Automated Analysis

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

#2 Multiple Feature Models



Two Key Requirements



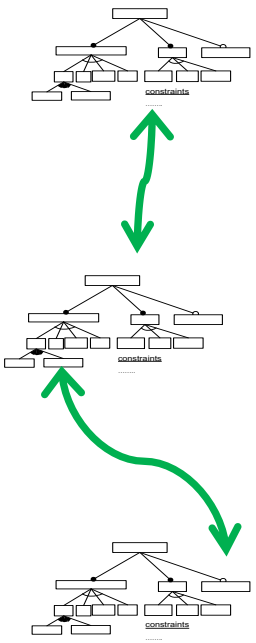
- **#1 Automated analysis**
  - Aka support to better understand and play with your feature model (TVL model)

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

David Benavides *, Sergio Segura, Antonio Ruiz-Cortés

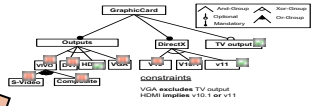
- **#2 Managing <u>multiple</u> feature models**
  - Composing / Decomposing / Diff and Reasoning about their relationships
  - Combining these operators

FAMILIAR language and environment

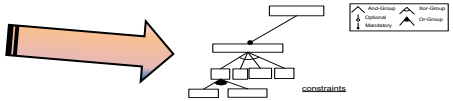



```
// foo.fml
fm1 = FM ("foo1.tvl")
fm2 = FM ("foo2.m")
fm3 = merge intersection { fm1 fm2 }
c3 = counting fm3
renameFeature fm3.TV as "OutputTV"
fm5 = aggregate { fm3 FM ("foo4.xml") }
assert (isValid fm5)
fm6 = slice fm5 including fm5.TV.*
export fm6
```

FAMILIAR

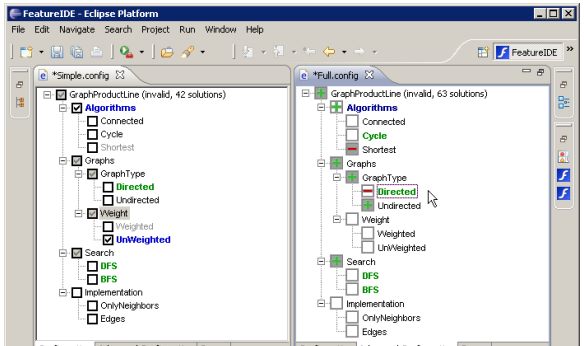


True/False
8759
"OutputTV", "TV"



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

eclipse

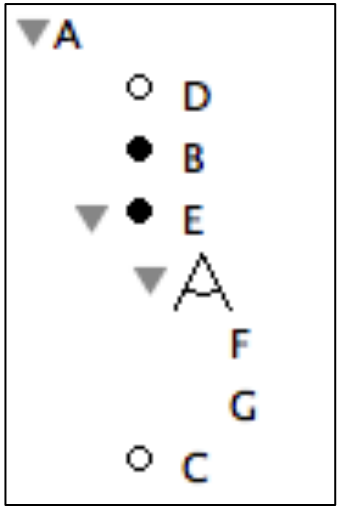


FAMILIAR … features

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>
```

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

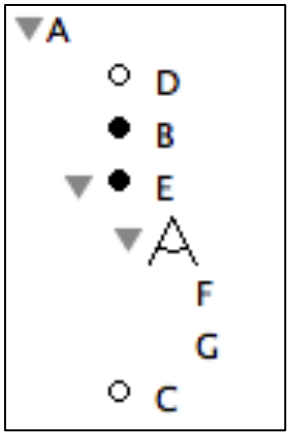
```
fm1 = FM ("foo1.tvl")
ft1 = root fm1
ft2 = fm1.B
fts = fm1.*
n = size fts
n2 = cores fm1

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

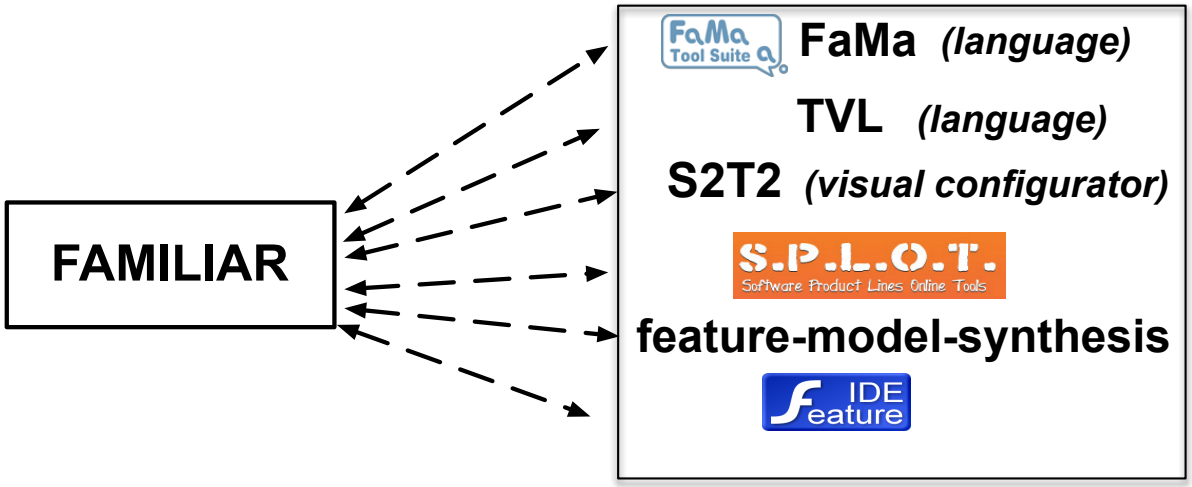
Typed language

```
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



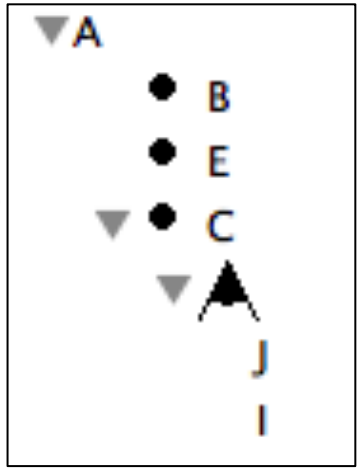
Internal notation *or* by “filename extensions”

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

ftAccessors.fml

Feature Accessors (1)

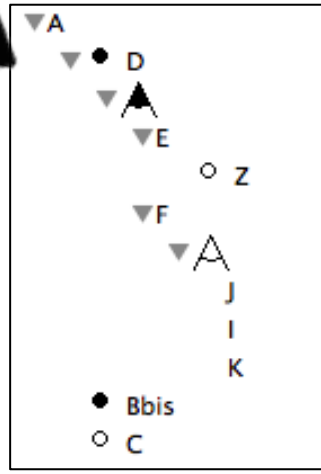
```
fm1 = FM (A : B [C] ; B : E F ; C : (I|J) ; )

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
```

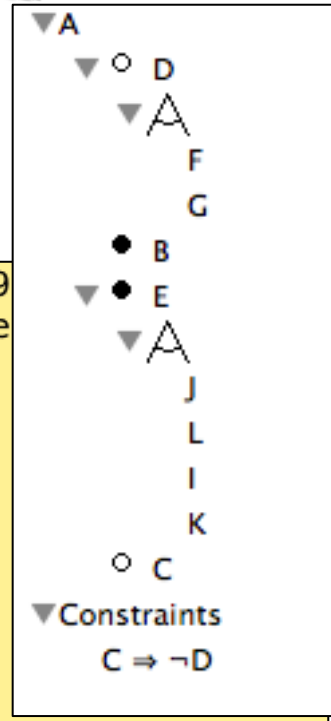



```
acher-scr:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar ftAccessors2.fml
Bbis

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!
```

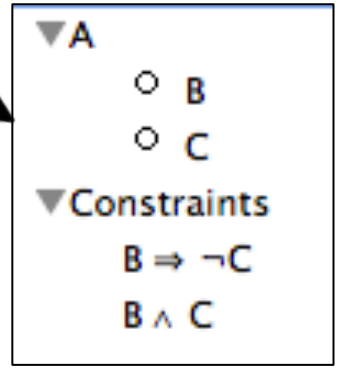



```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Re
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> cFM1
cFM1: (FEATURE_MODEL) A: 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
```

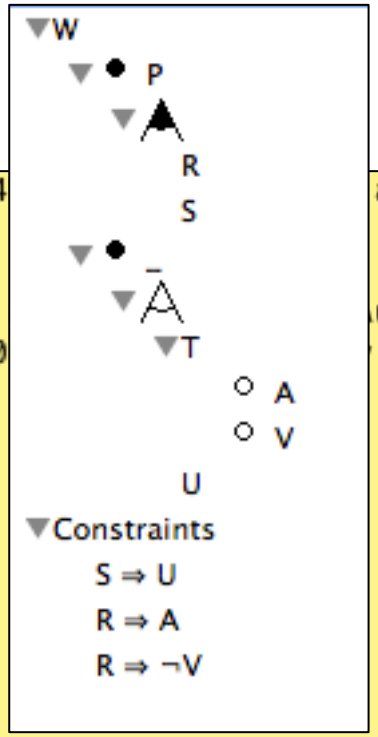



```
acher-scr:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar operatorsFM.f
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoni
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> ls
(FEATURE_MODEL)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



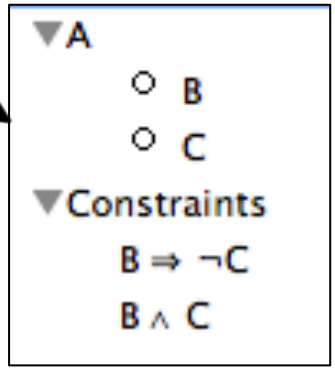

```
acher-scr:FML-scripts macher$ java -jar -Xmx1024                    ar operatorsFM2.fml
cores: {P;W}

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

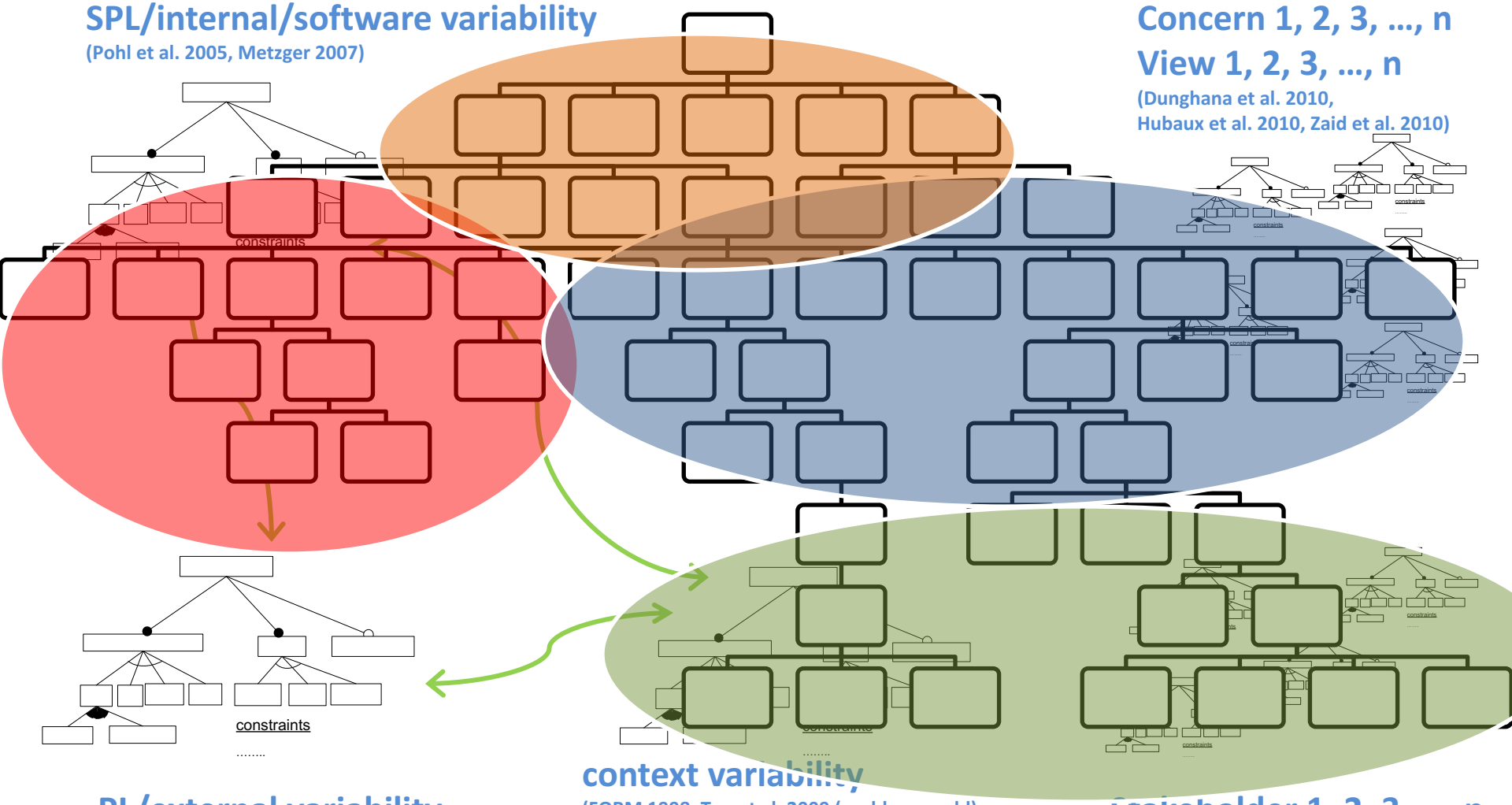
Operations for Feature Models (3)

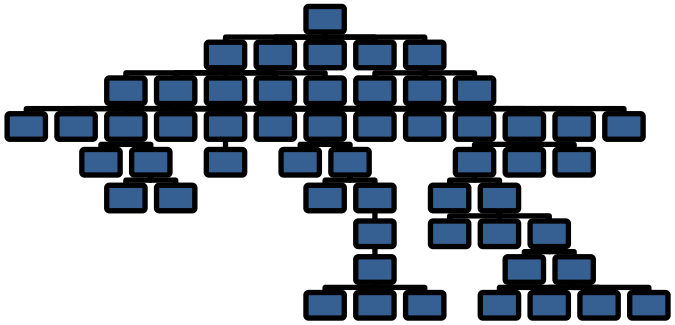
```
fm1 = FM (A : [B] [C] ; B -> !C ; B and C ; )
b1 = isValid fm1

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
```

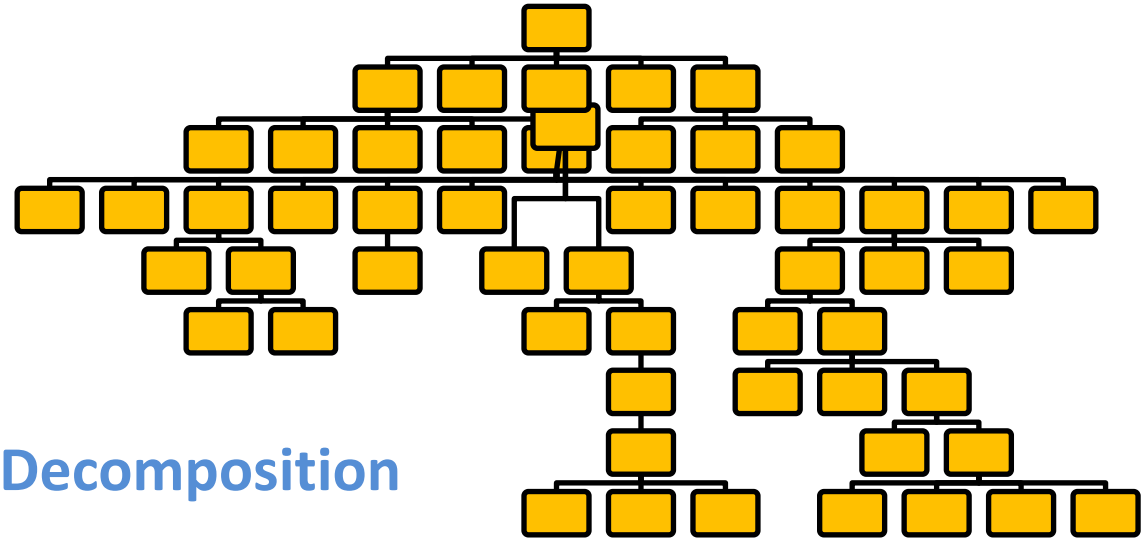



```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar operatorsFM3.fml
removing constraint... (B & C)

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

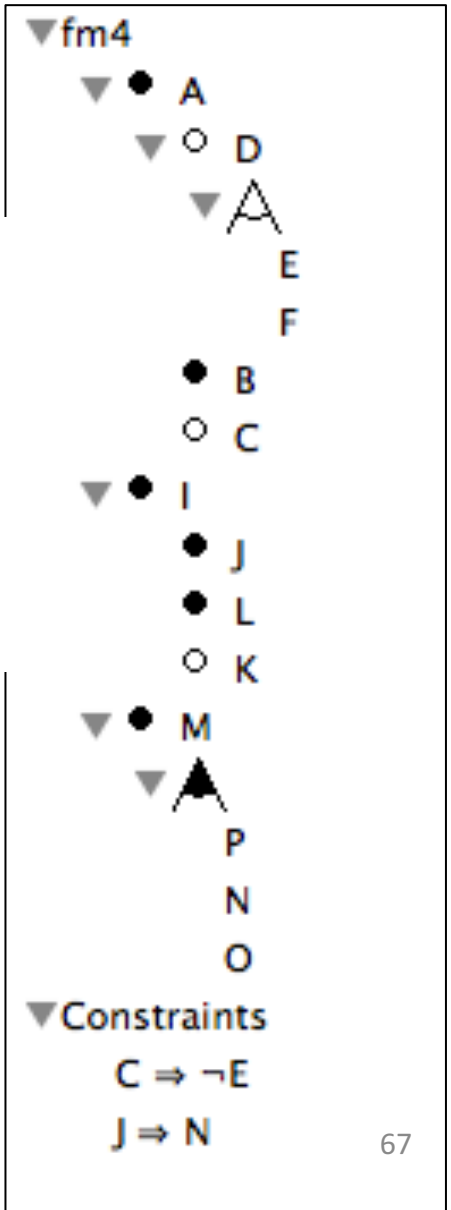
feature models

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

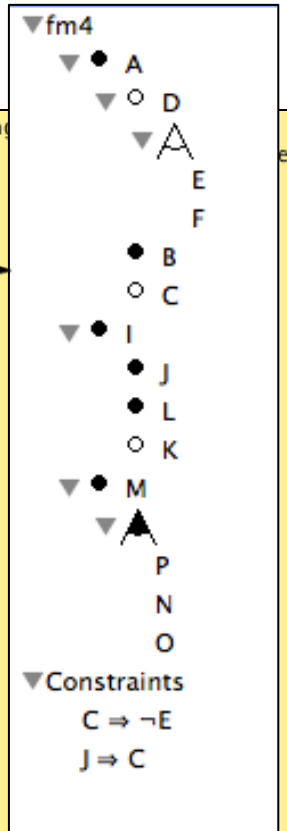
Composing Feature Models (2)

```
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
```

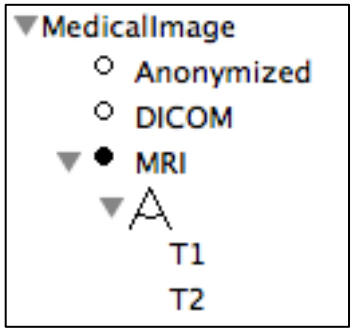
Previous version

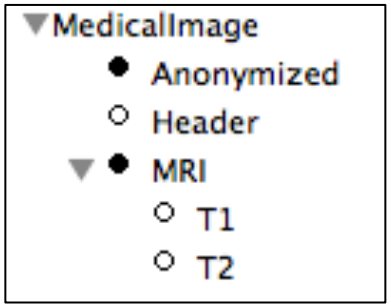
```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar ag
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> cores fm4
res0: (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);
```

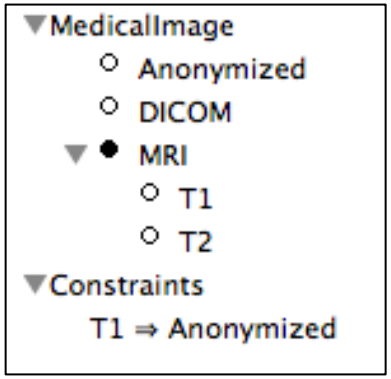


mergeMI.fml

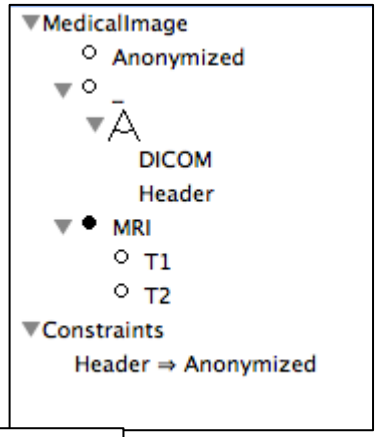
Merging Feature Models

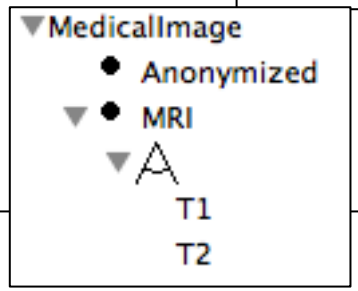







```
fmsupp1 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : (T1|T2) ;  )
fmsupp2 = FM (MedicalImage : Anonymized MRI [Header] ; MRI : [T1] [T2] ; )
fmsupp3 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : [T1] [T2] ; T1 -> Anonymized; )

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
```



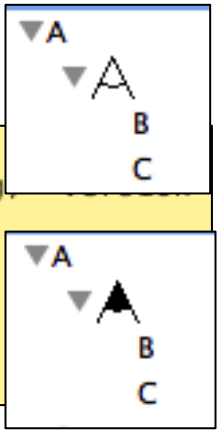


compare.fml

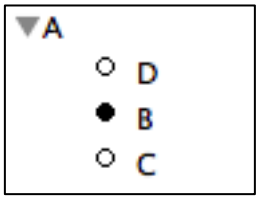
Comparing Feature Models

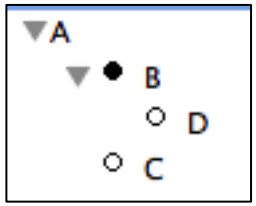
```
fm0 = FM (A: (B|C) ; )
fm1 = FM (A: (B|C)+ ; )
```

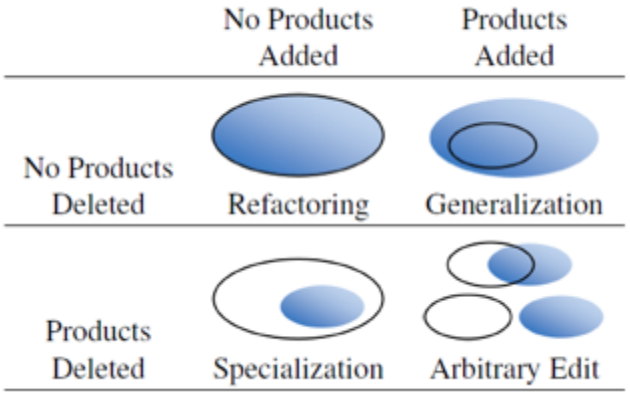
```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> cmp23
cmp23: (STRING)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

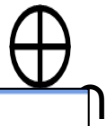
Merge Intersection: Available Suppliers

- 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

⊕ ∪

Yes!

Can suppliers provide *all* products?

“compare”

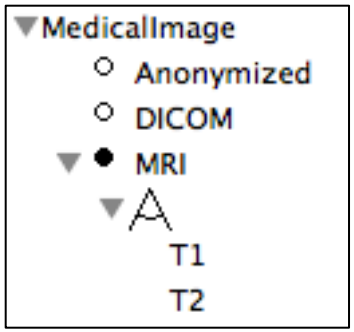
Registration
- Header
  - v1
- Modality
  - CT
  - MRI
- Format
  - DICOM
  - Nifti
- Constraints
  - DICOM ⇒ ¬MRI

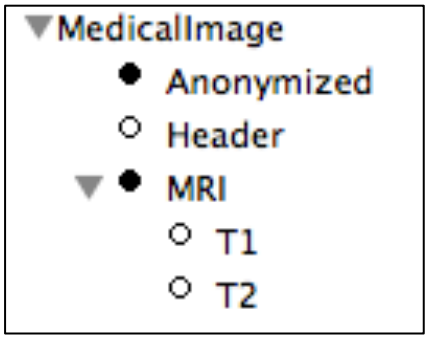
suppliersExample.fml

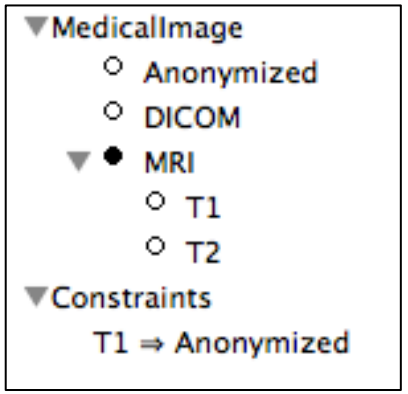
In FAMILIAR

```
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)
```

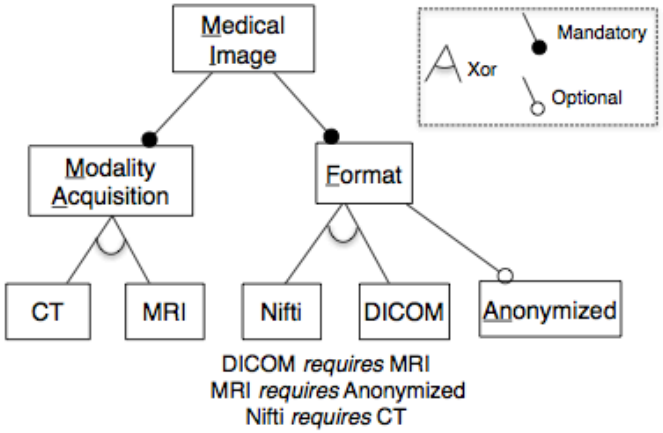
Anonymized v Header v DICOM v ~T1 v ~T2

Merging operation: semantic issues (2)

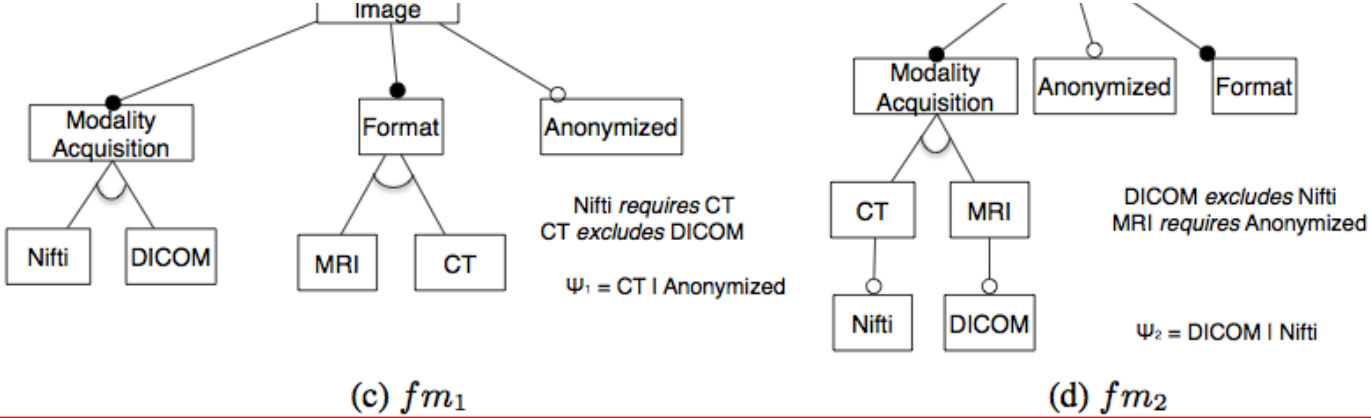
φ

$$s_0 = \{$$
$$\{MI, MA, F, CT, Nifti\},$$
$$\{MI, MA, F, CT, Nifti, AN\},$$
$$\{MI, MA, F, DICOM, MRI, AN\}$$
$$\}$$

Union
Intersection
Diff



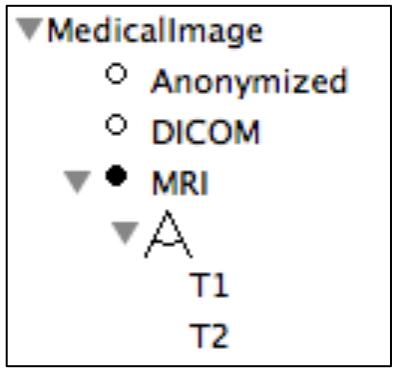
How to synthesise a feature model that represents the union of input sets of configurations?



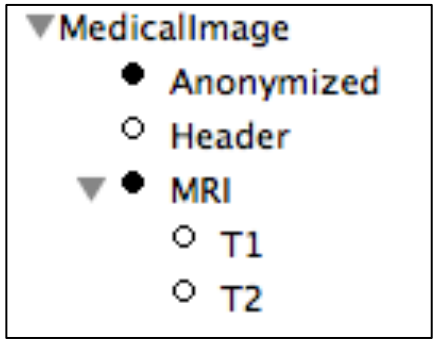
(c) fm_1 (d) fm_2

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

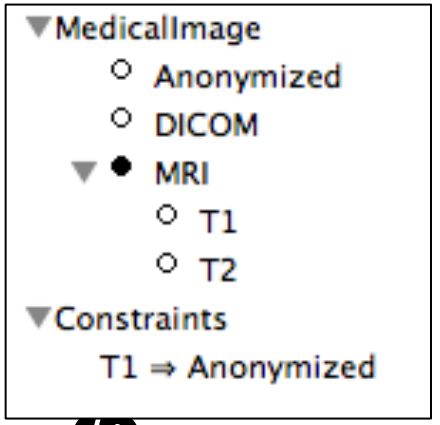
Merging operation: algorithm



φ_1



φ_2



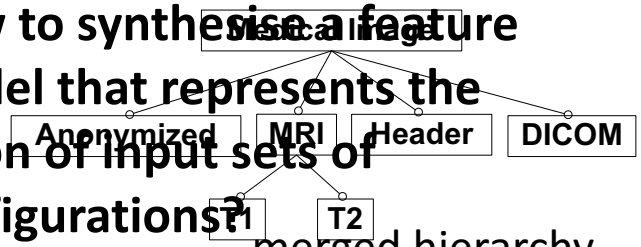
φ_3

φ_{123}

merged propositional formula

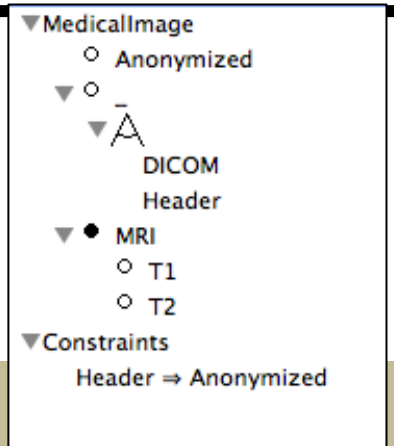
+

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



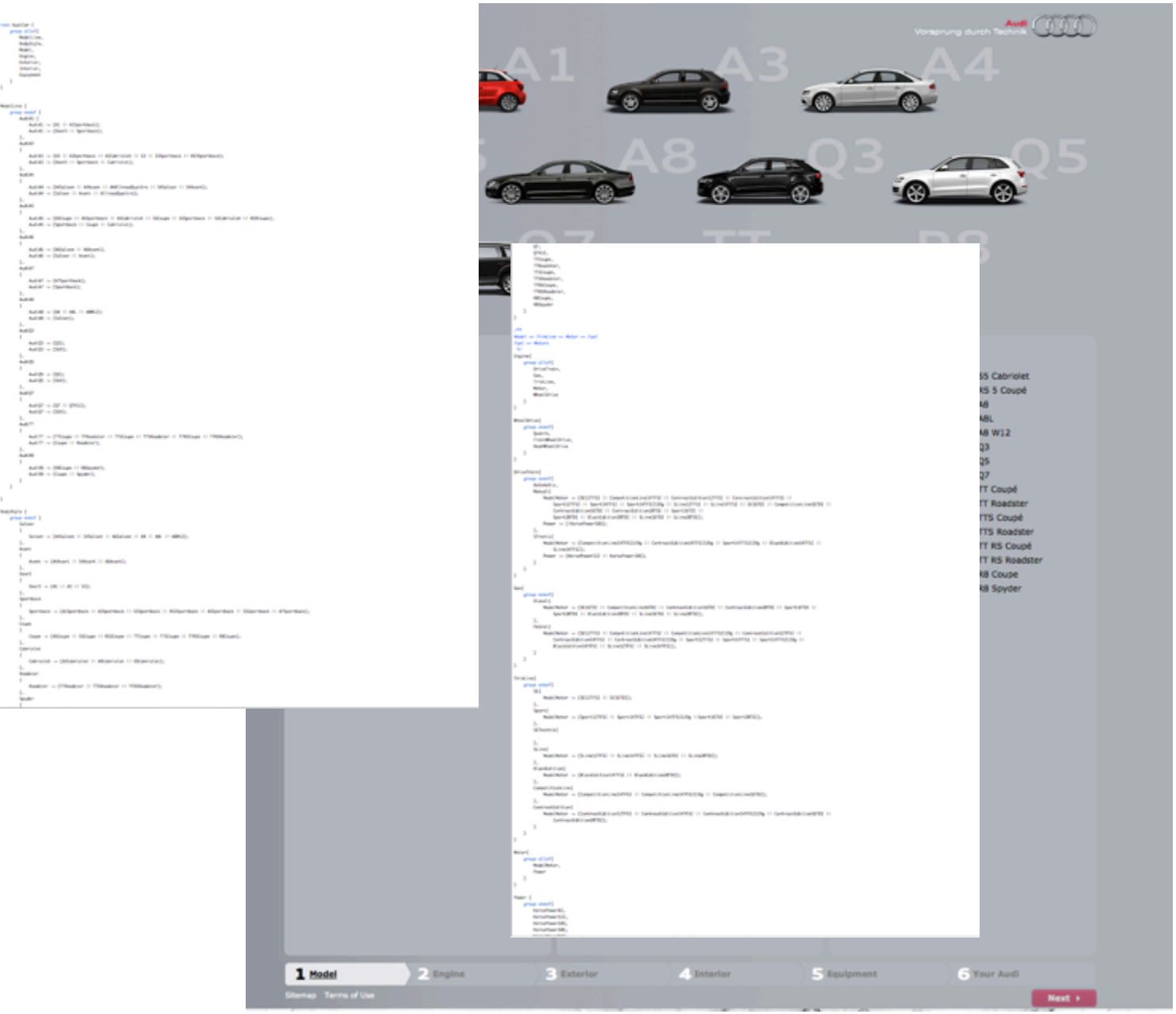
merged hierarchy

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



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

Building “views” of a feature model

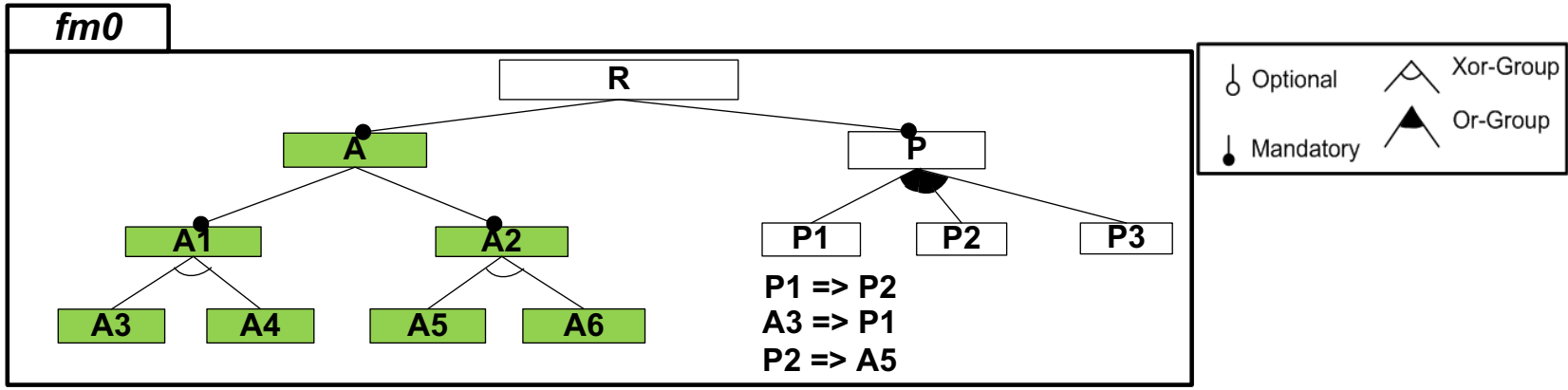


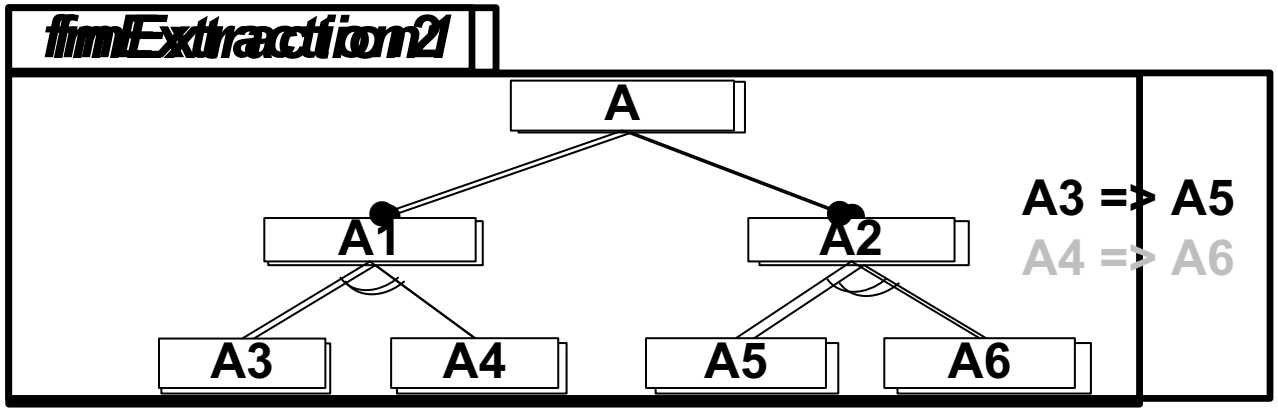
Building “views” of a feature model

- 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



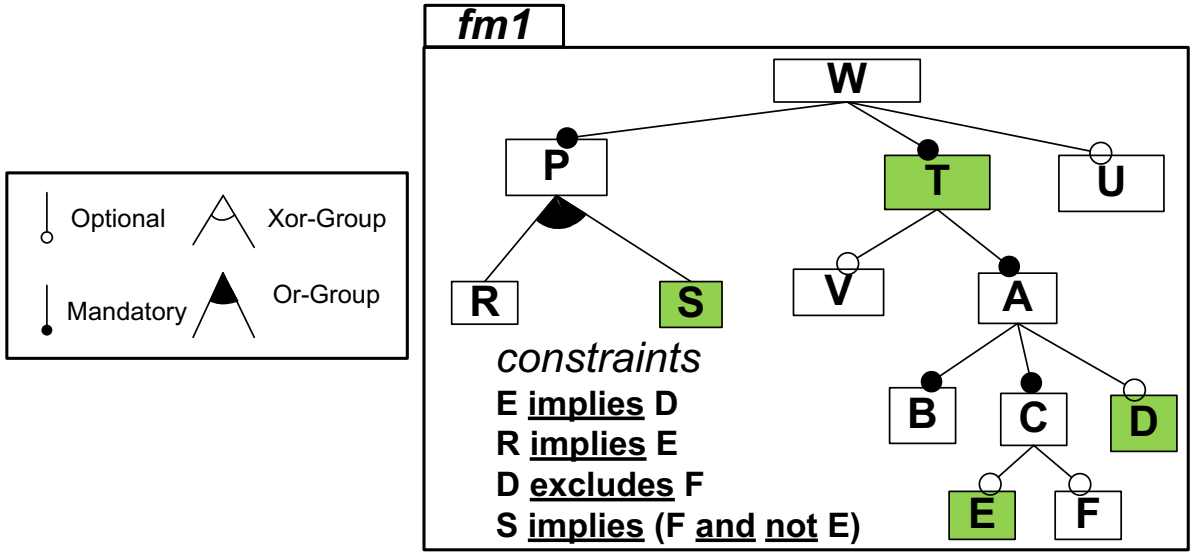


Problem: You can select **A3** without **A5**

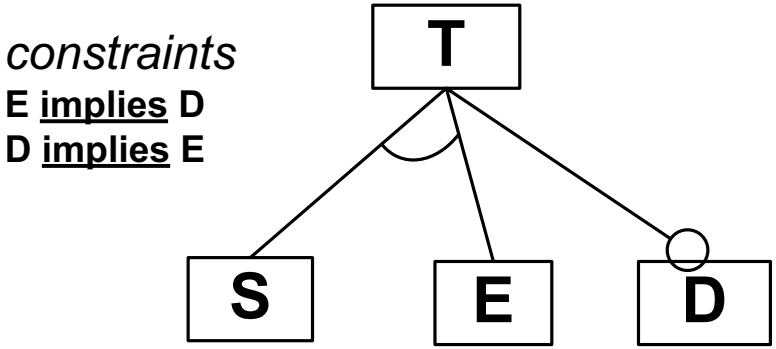
Hierarchy and Configuration matter!

Slicing Operator

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

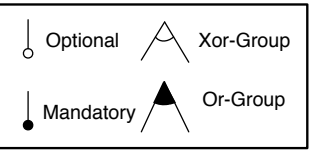


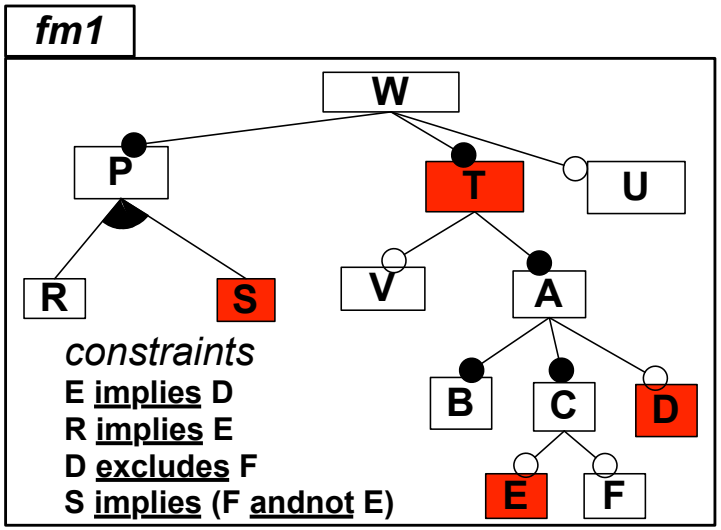
slice : a new feature model, representing a projected set of configurations



Slicing operator: going into details

projected set of configurations



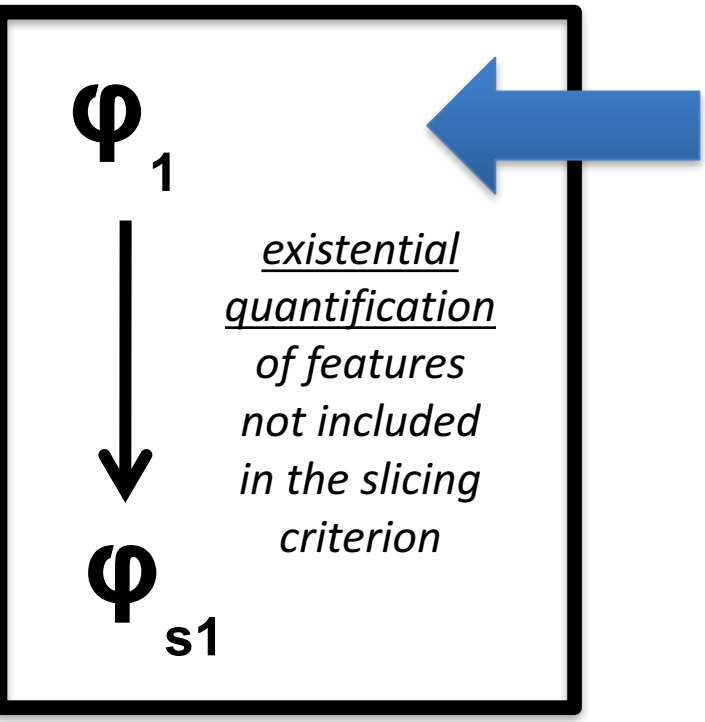


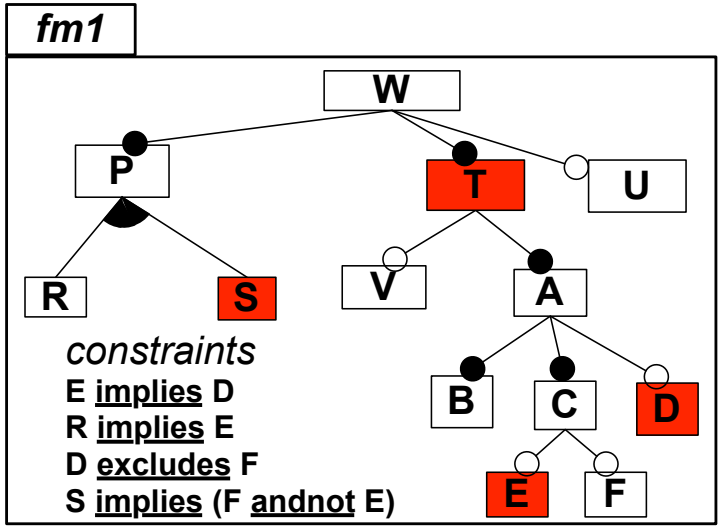
fm1 = {
{A,B,C,D,E,P,R,T,U,W},
{A,B,C,E,P,R,S,U,W},
{A,B,C,D,E,P,R,T,W},
{A,B,C,E,P,R,S,T,W},
{A,B,C,E,P,R,S,U,W},
{A,B,C,E,P,R,S,T},
{A,B,C,D,E,P,R,T,W},
} }

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

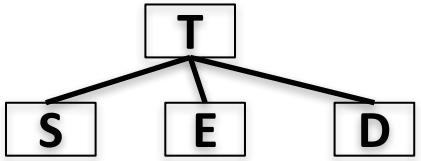
Slicing operator: going into details

synthesizing the corresponding feature model

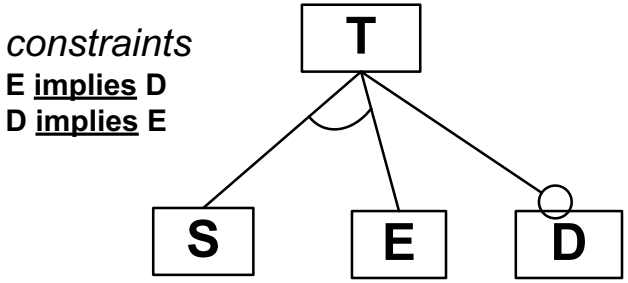




+



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

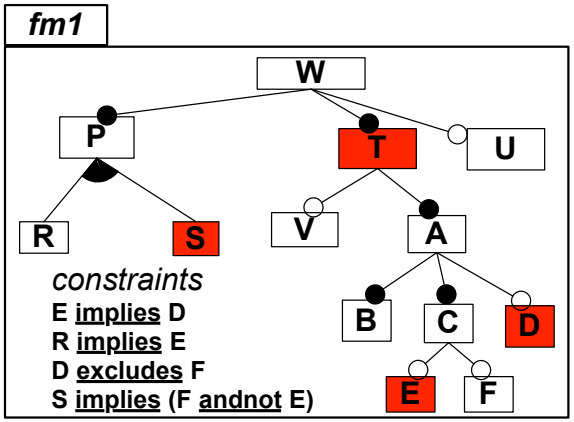


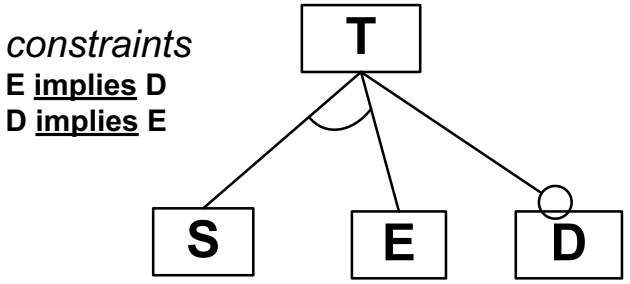
see also [Acher et al., ASE'11/AOSD'12]

slicingOp2.fml

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

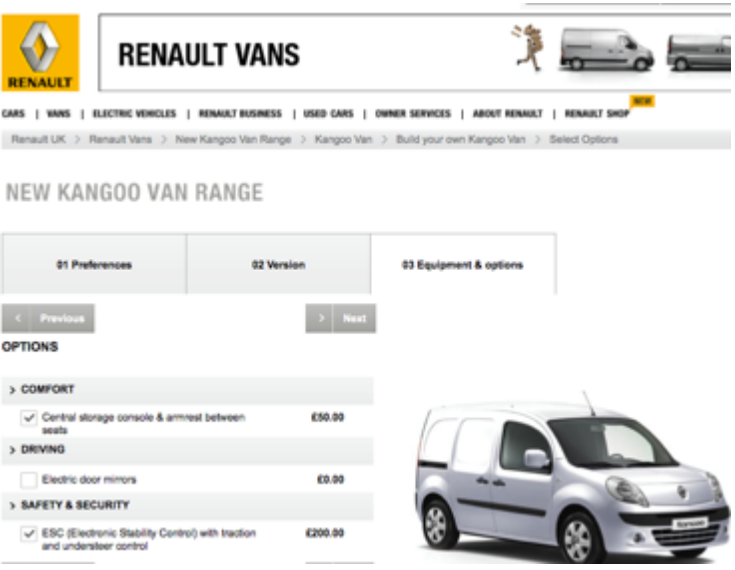
Slicing with FAMILIAR (2)

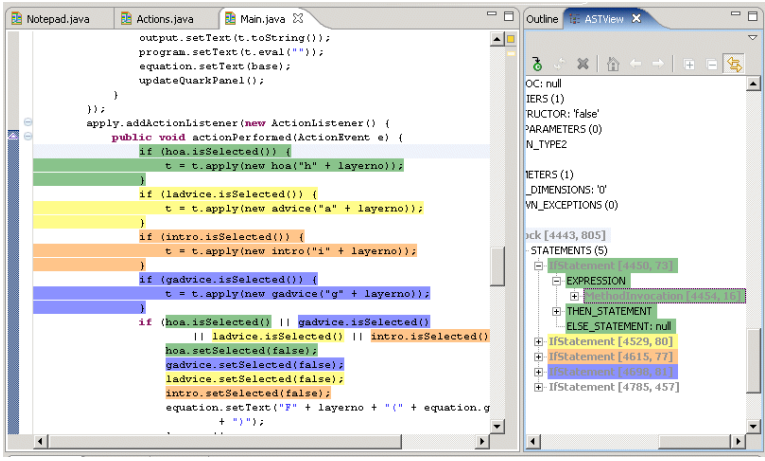
```
fm1 = FM (W : P T [U] ; T : [V] A ;
                          A : B C [D] ;
                           C : [E] [F] ;
                           P : (R|S)+ ;
                           E implies D ; R implies E ;

                           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
```

Putting all together: Example 2

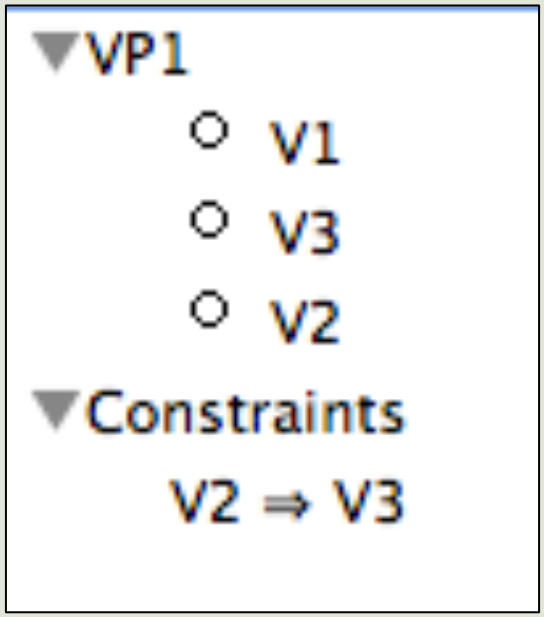
From marketing, customers, product management

From existing software assets (technical variability)





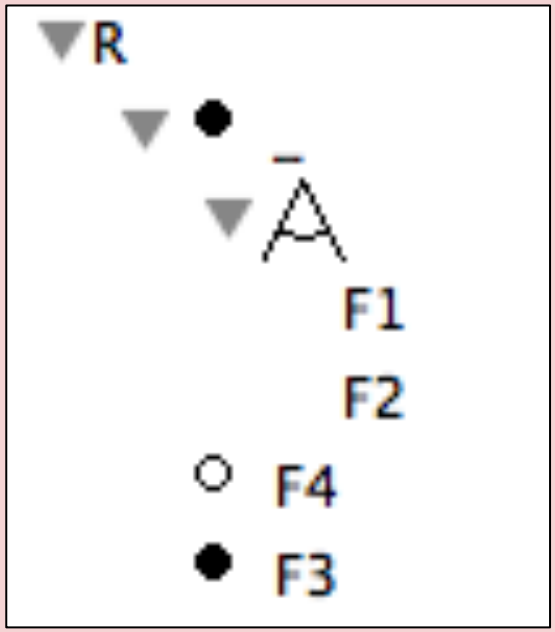
Metzger, Heymans et al. "Disambiguating the Documentation of Variability in Software Product Lines: A Separation of Concerns, Formalization and Automated Analysis" (RE'07)



usefulness

V1 ⇔ f1
V2 ⇔ f2
V3 ⇔ f3

realizability

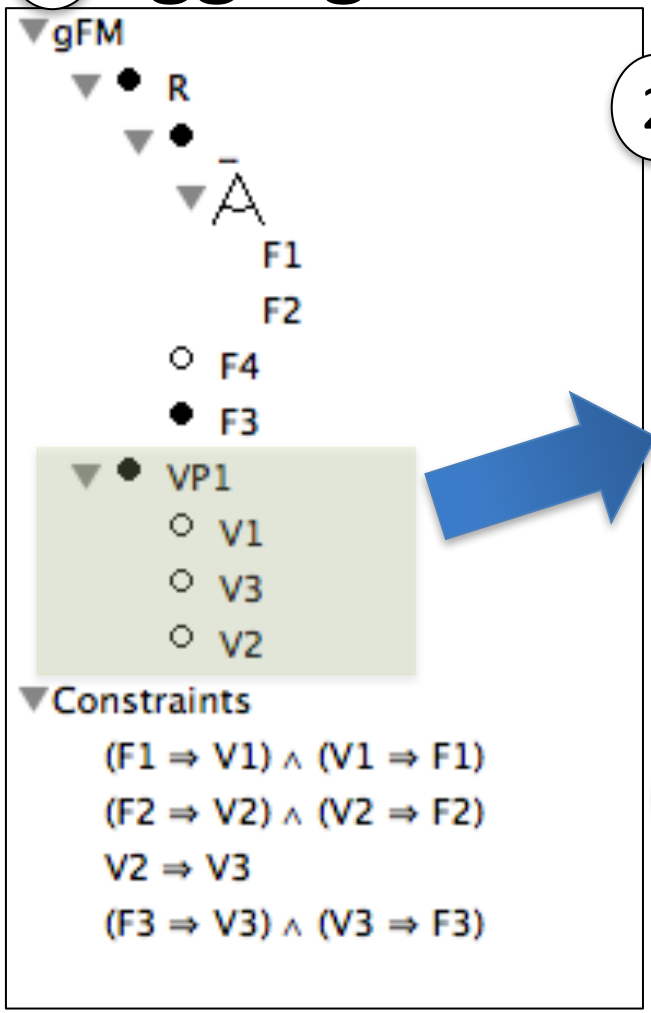


From marketing, customers, product management

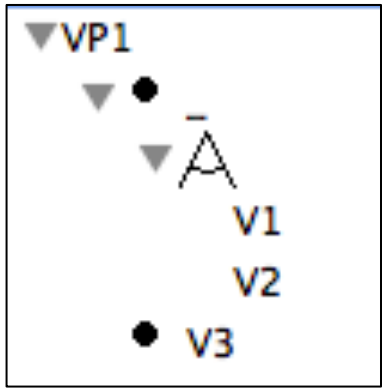
From existing software assets

Realizability checking

1 **aggregate**



2 **slice** (“realizable part”)



φ

3 **compare**

{{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
*/
```

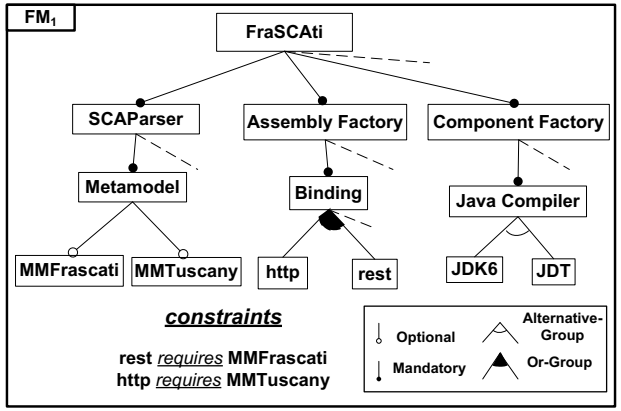
```
fmSoftware = FM (R : (F1|F2) F3 [F4] ; )
```

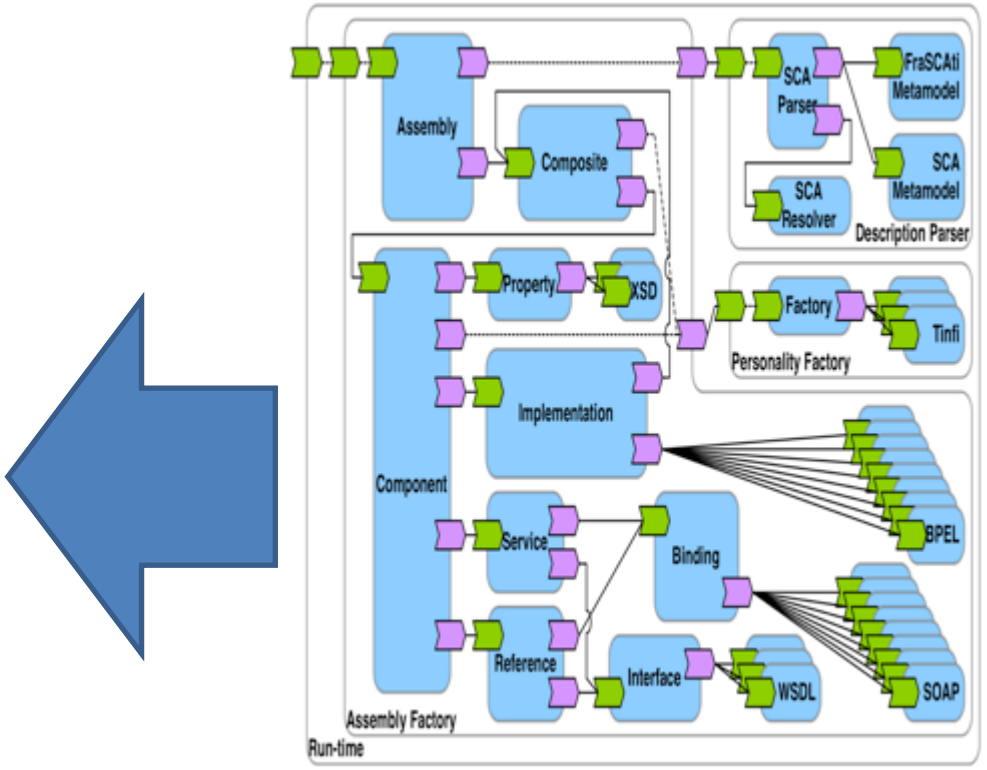
```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.6.jar realizability.fml
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)  version 0.9.9.6
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
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}}
```

Putting all together: Example 3

#1 Reverse Engineering Architectural Feature Models

Case Study: FraSCAti Architecture

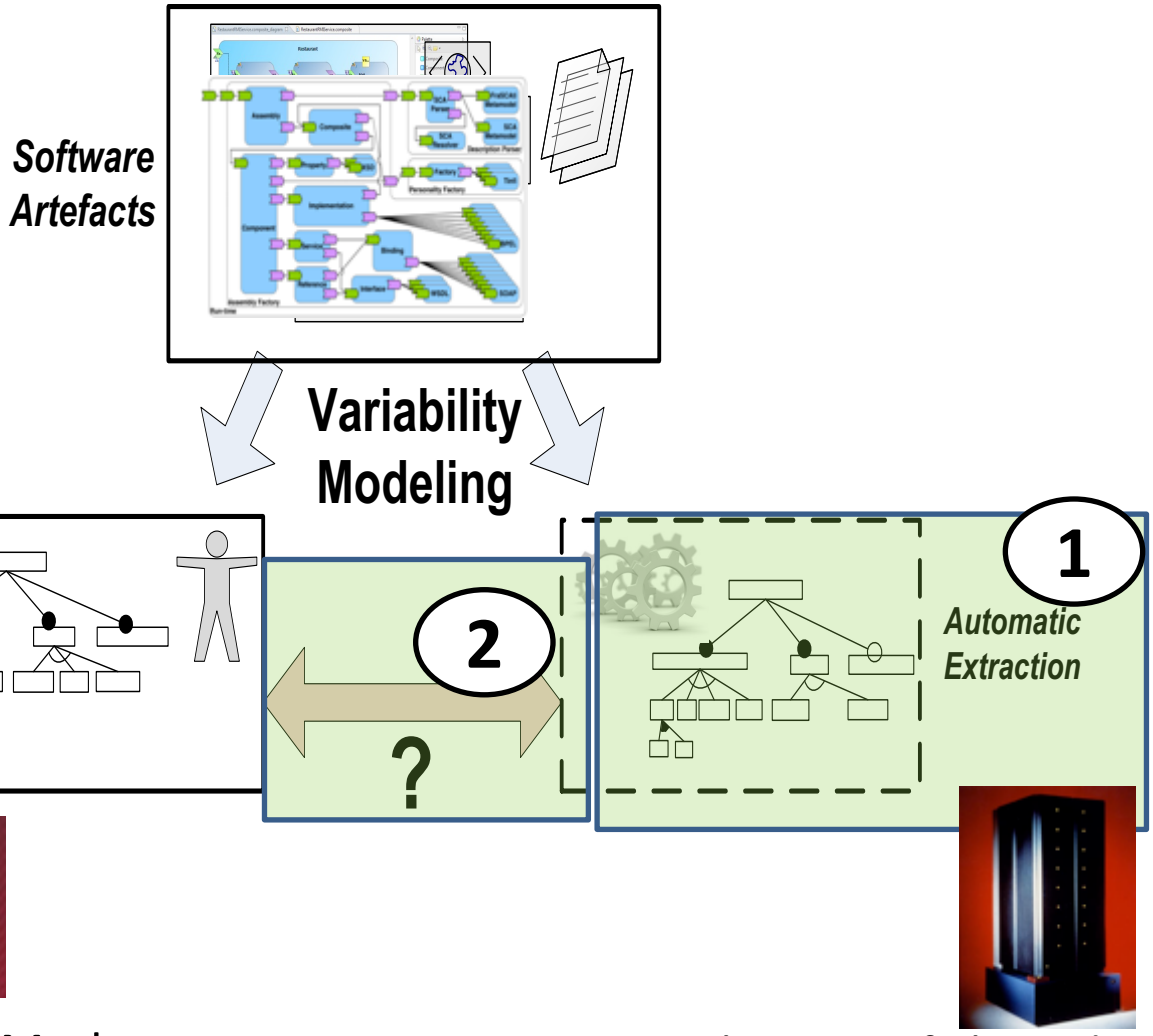




[Acher et al., ECSA'11]
[Acher et al., BENEVOL'11]
[Acher et al., GDR GPL'12]

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

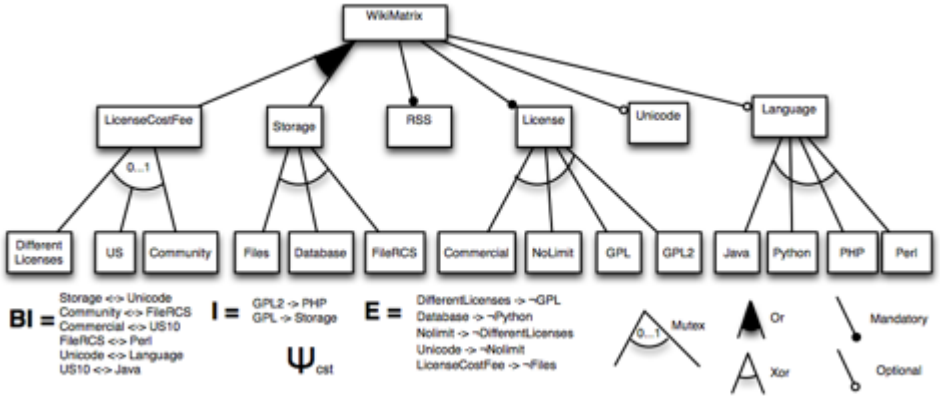
Highlights

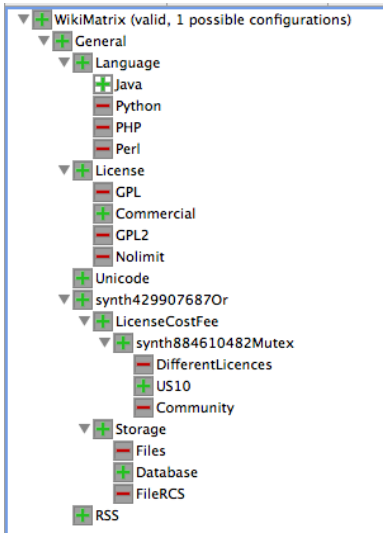
- Automated Procedure
  - Extracting and Combining Variability Sources

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

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

#2 from product descriptions to feature models



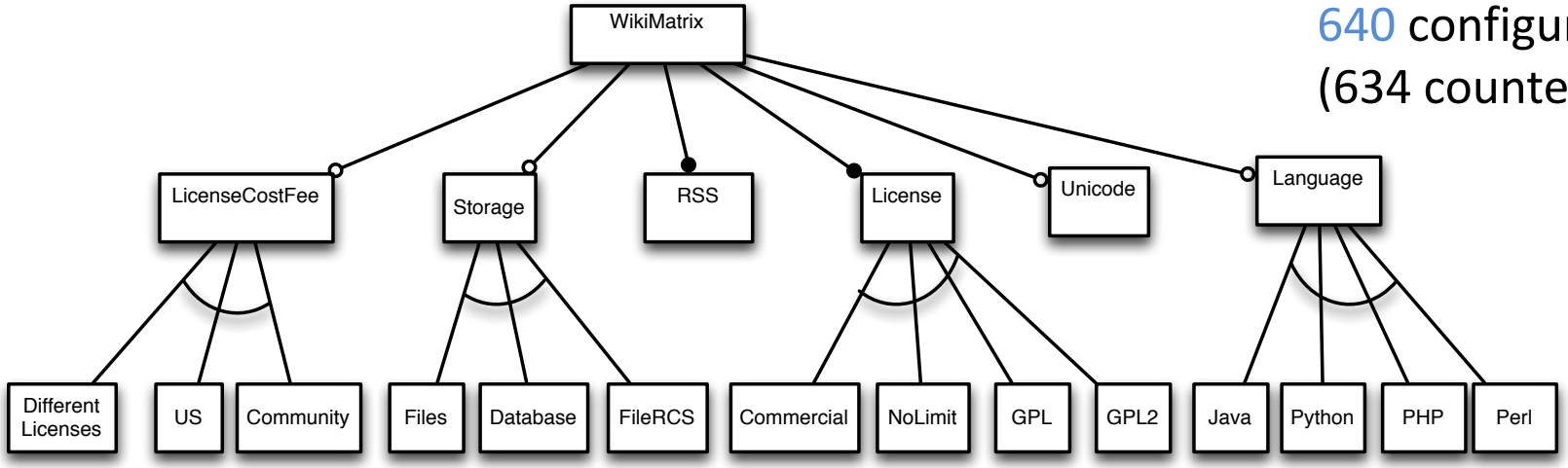


[Acher et al., VaMoS'12]

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

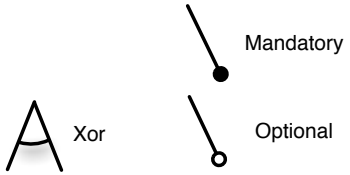
Manual extraction of a feature model from product description(s) is not possible

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

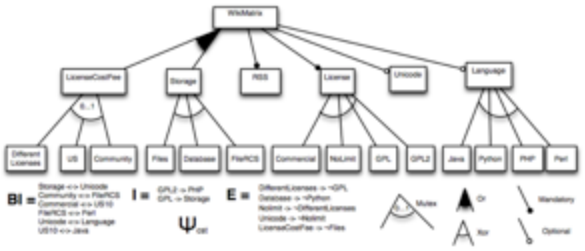


Automation

- 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

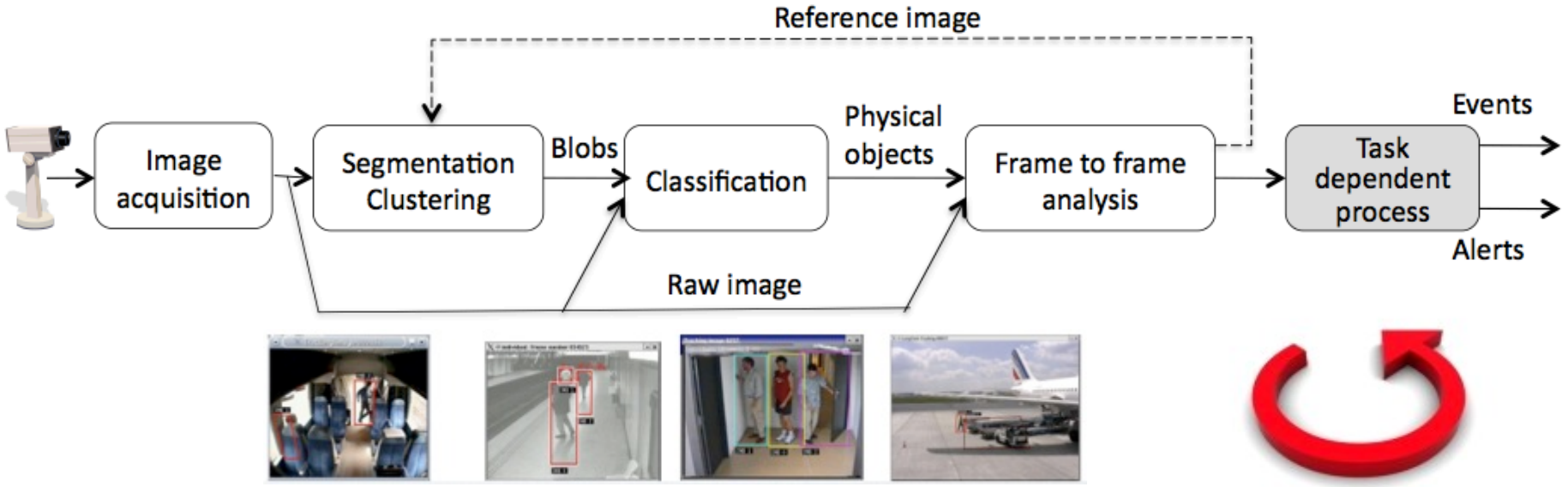
- Feature models {fm1, fm2,…,fm8} are **merged**
  - Output: a new feature model
    - Configuration: union of input sets of configurations
    - Hierarchy: by default, we exploit the structure of the tabular data
      - Can be overridden by specific user directive
  - VariCell
    - DSL built on top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



Putting all together: Example 4

Modeling Variability From Requirements to Runtime

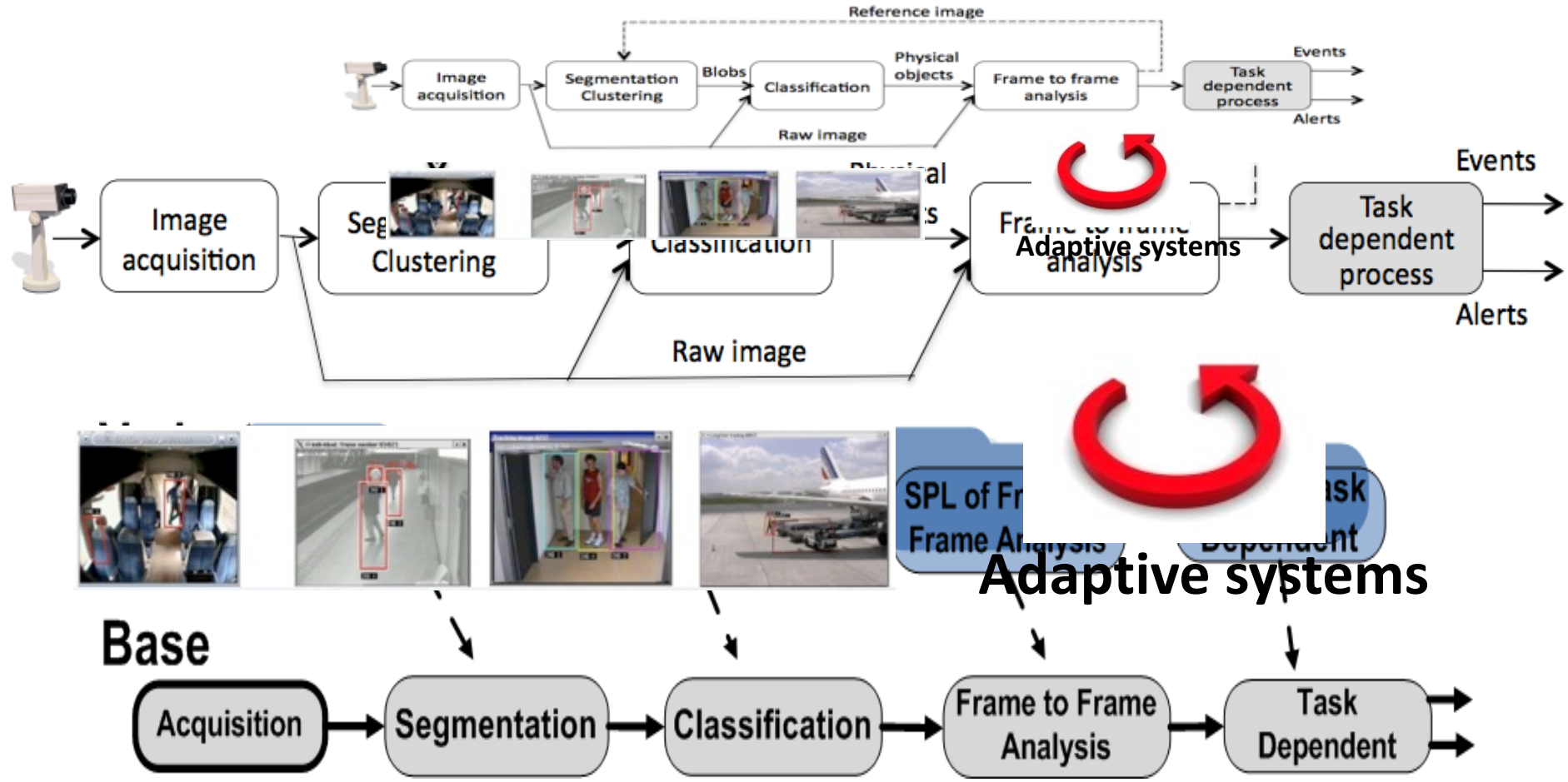
The case of video surveillance processing chains



Adaptive systems

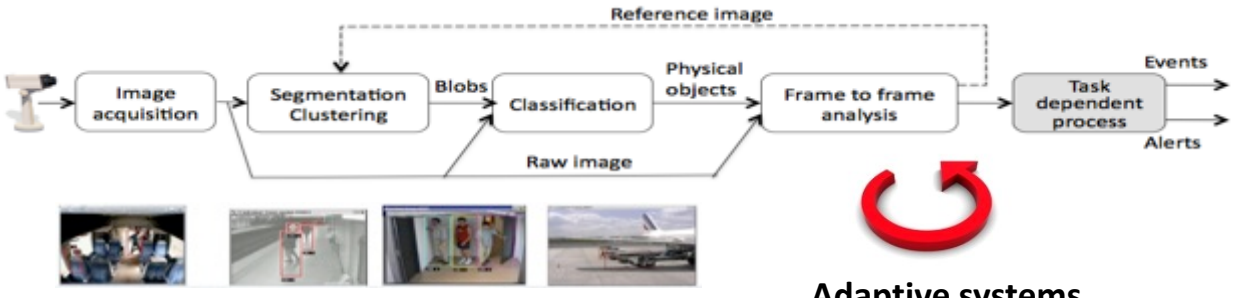
Collaboration with Sabine Moisan and Jean-Paul Rigault (INRIA)

Video surveillance processing chains



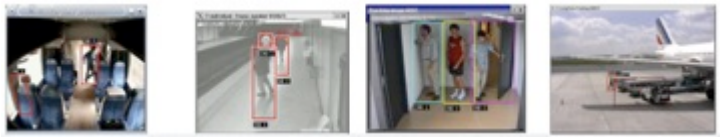
large number of software configurations for a large number of requirements

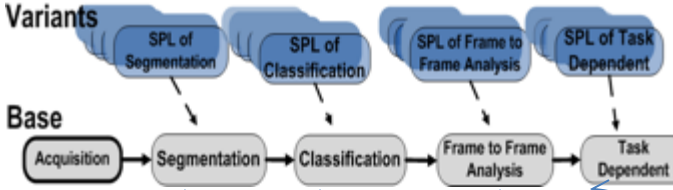
Video surveillance processing chains

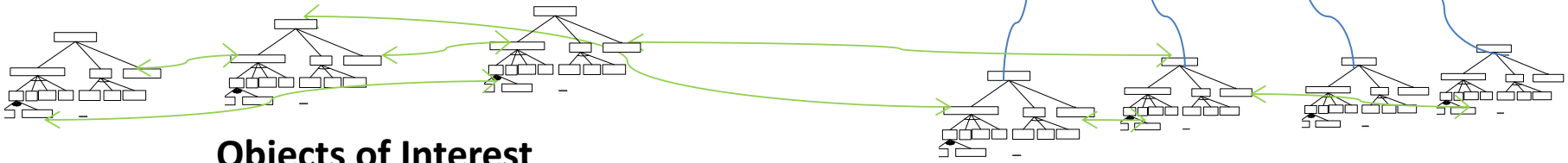


Adaptive systems

Software Product Line (SPL) approach







Scene Context

Objects of Interest

Specific Task

Video Surveillance Application Requirements → **Software Platform Configurations**

Implementation: under the hood