

Modeling Variability

(introduction to feature models)

Mathieu Acher

Maître de Conférences

mathieu.acher@irisa.fr

Material

[https://github.com/FAMILIAR-project/
HackOurLanguages-SIF](https://github.com/FAMILIAR-project/HackOurLanguages-SIF)

Plan

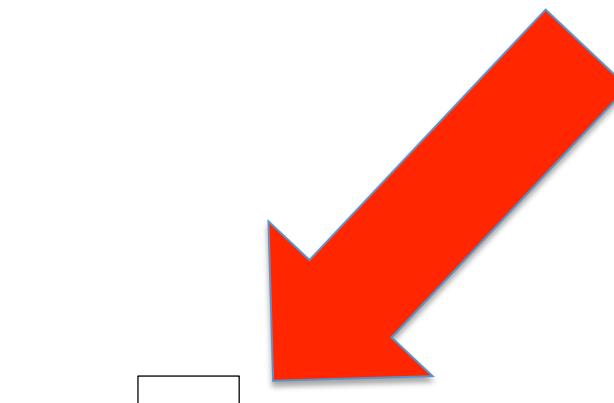
- Challenges and Overview
 - Developping billions of software product is hard but now a common practice
- Implementing Variability
 - Revisit of existing techniques and curriculum
- Specificity of Product Line Engineering
 - Process, methods
- Feature Models
 - **Defacto standard for modeling product lines and variability**
 - **Syntax, semantics, automated reasoning, synthesis**

Contract

- The idea of software product lines and variability
 - You will be able to recognize this class of systems
 - Aware of the complexity, the specific development process, and existing techniques
- **Feature modeling**
 - **A widely used formalism for modeling product lines and configurable systems in a broad sense**
- Composing/Decomposing feature models with a domain-specific language
- Reverse engineering variability models

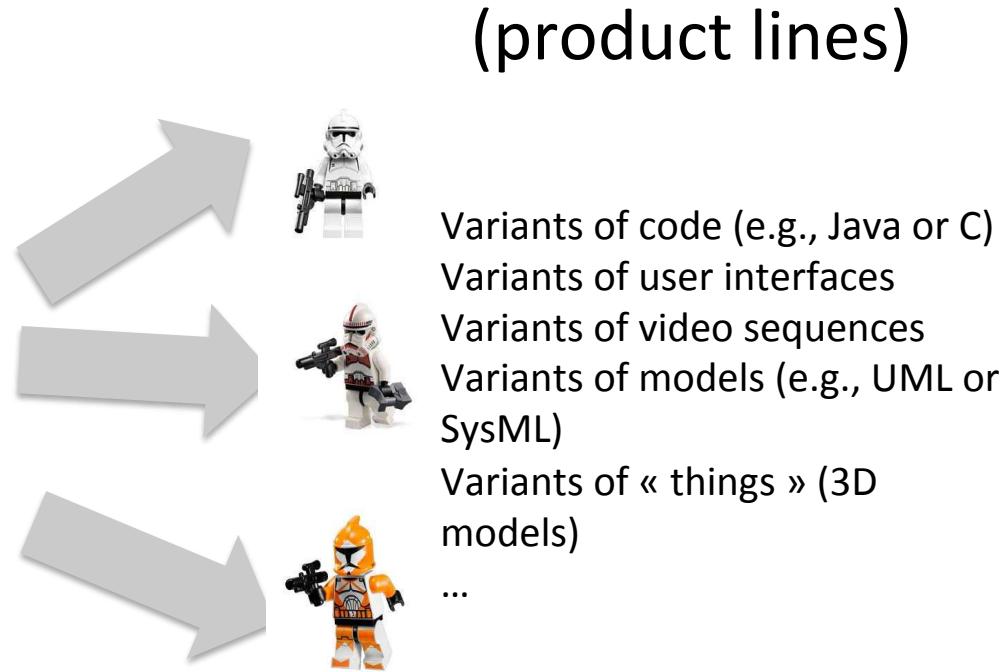
Modeling Variability

Modeling and Reverse Engineering Variability

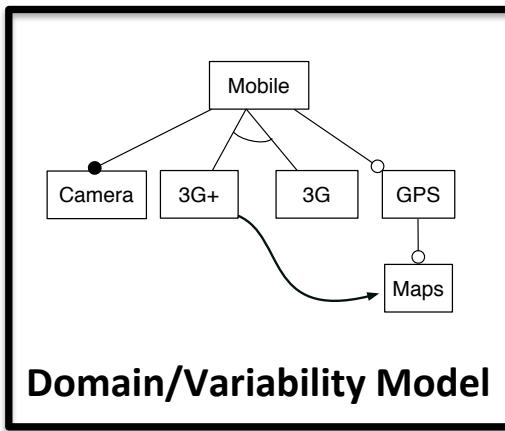


Feature models
or Product Matrices

Product	License	Price	Language Support	Language	WYSIWIG
W1	Commercial	10	Yes	Java	Yes
W2	NoLimit	20	No		Yes
W3	NoLimit	10	No		Yes
W4	GPL	0	Yes	Python	Yes
W5	GPL	0	Yes	Perl	Yes
W6	GPL	10	Yes	Perl	Yes
W7	GPL	0	Yes	PHP	No
W8	GPL	10	Yes	PHP	Yes



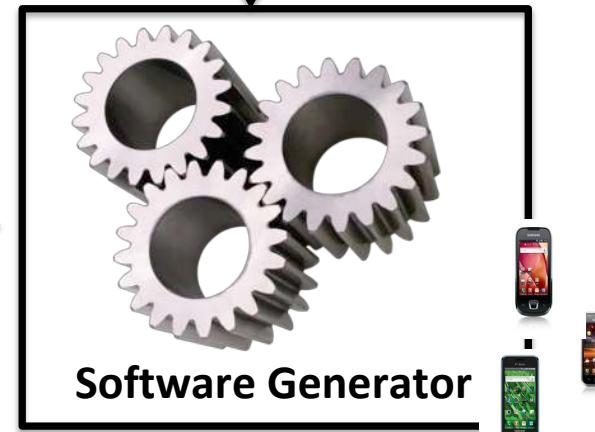
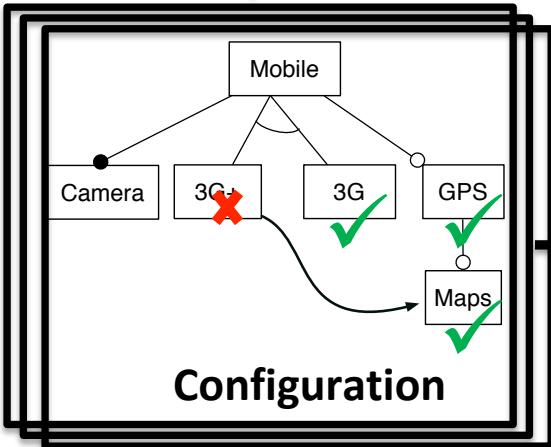
Domain Engineering



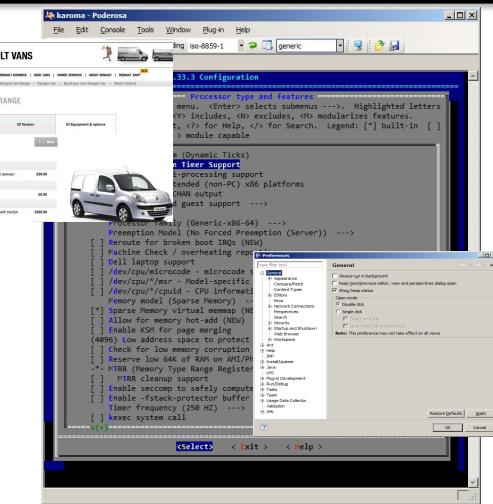
A screenshot of a software interface showing code in Notepad.java and an AST View panel. The code includes logic for "apply" and "intro" methods, dealing with device selection and layer numbers.

Domain Artefacts

Application Engineering

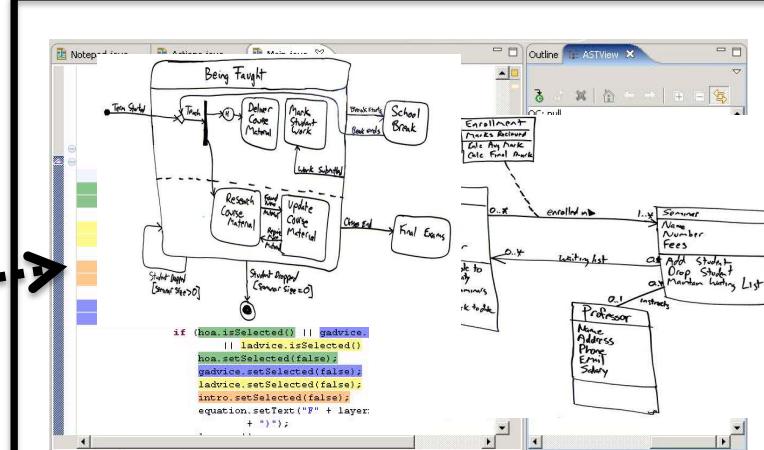


« the investments required to develop the reusable artifacts during **domain engineering**, are outweighed by the benefits of deriving the individual products during **application engineering** »



Variability Model

mapping



Base Artefacts (e.g.,
models)



Configuration



Software Generator
(derivation engine)



Quizz: what are the constraints over WORLD and BYE?

```
#include <stdio.h>

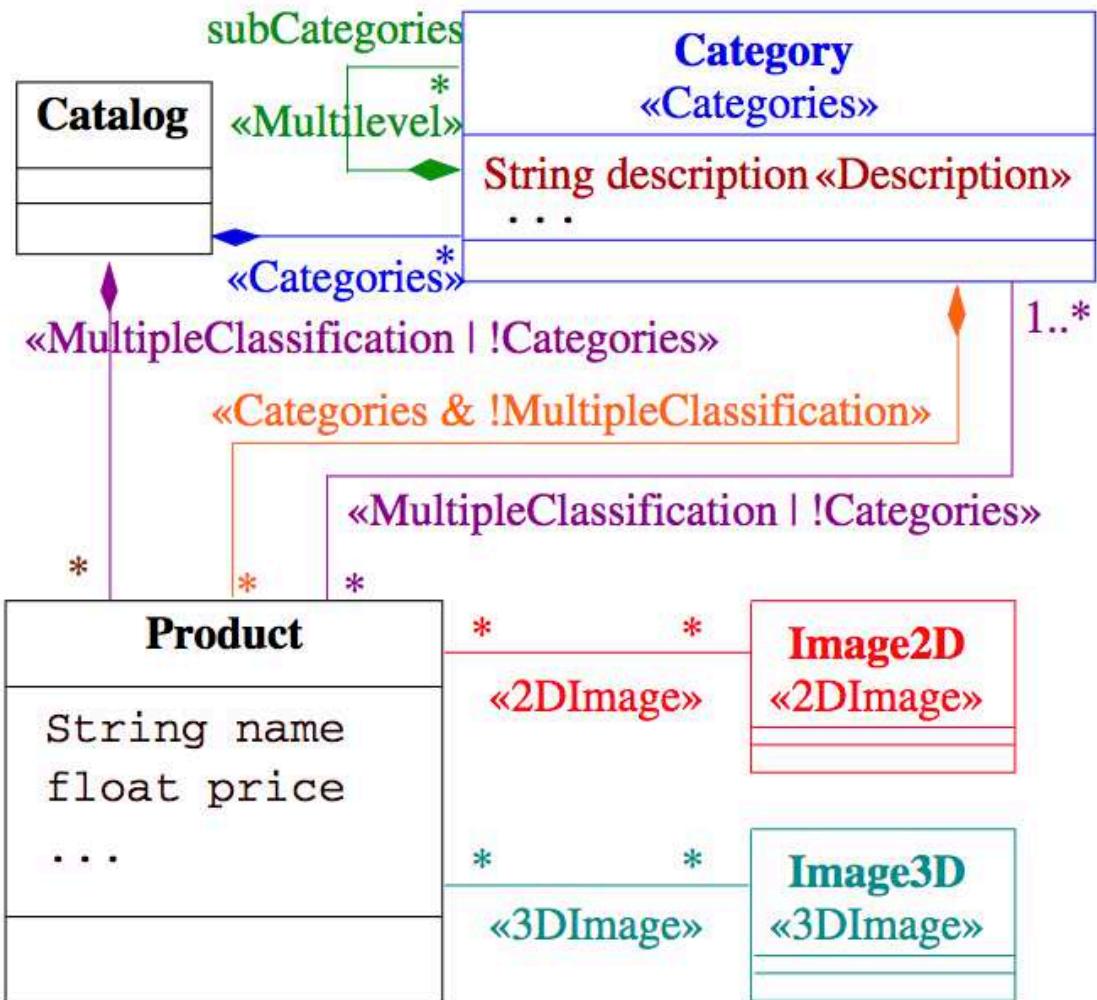
#ifndef WORLD
char * msg = "Hello_World\n";
#endif

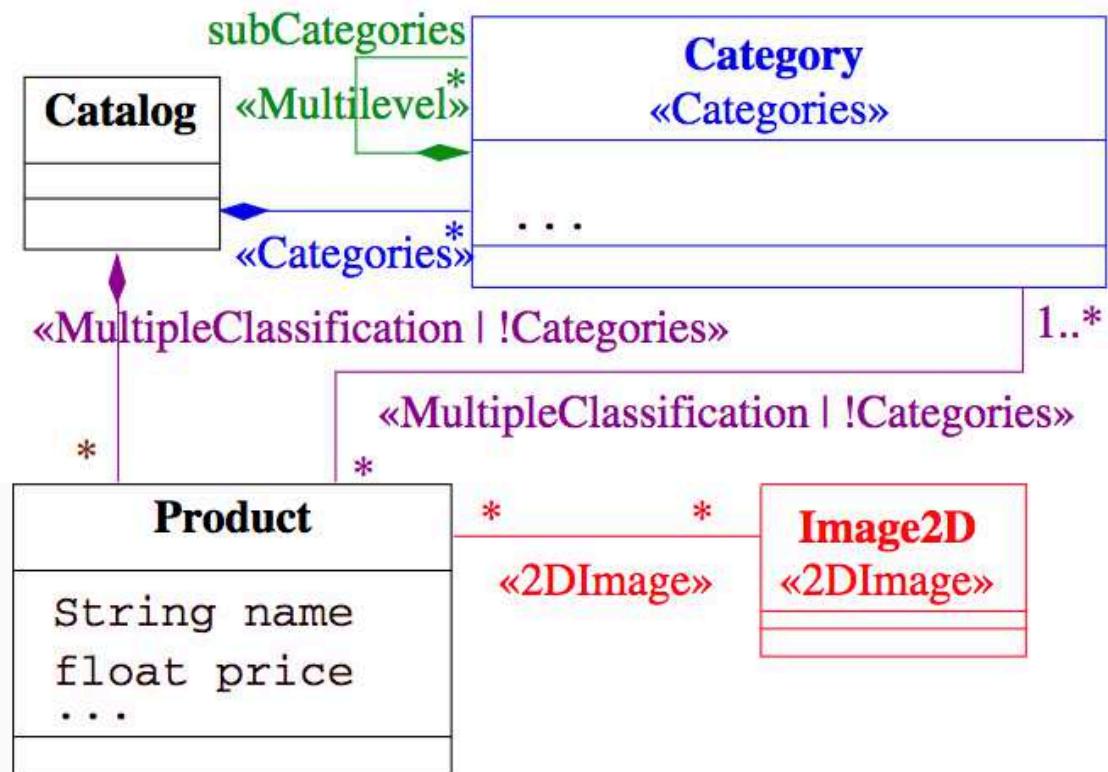
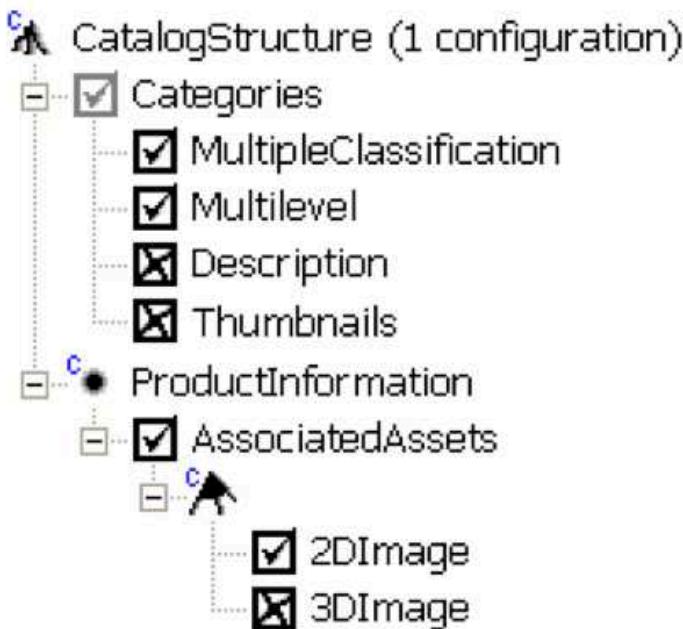
#ifndef BYE
char * msg = "Bye_bye!\n";
#endif

main() {
    printf(msg);
}
```

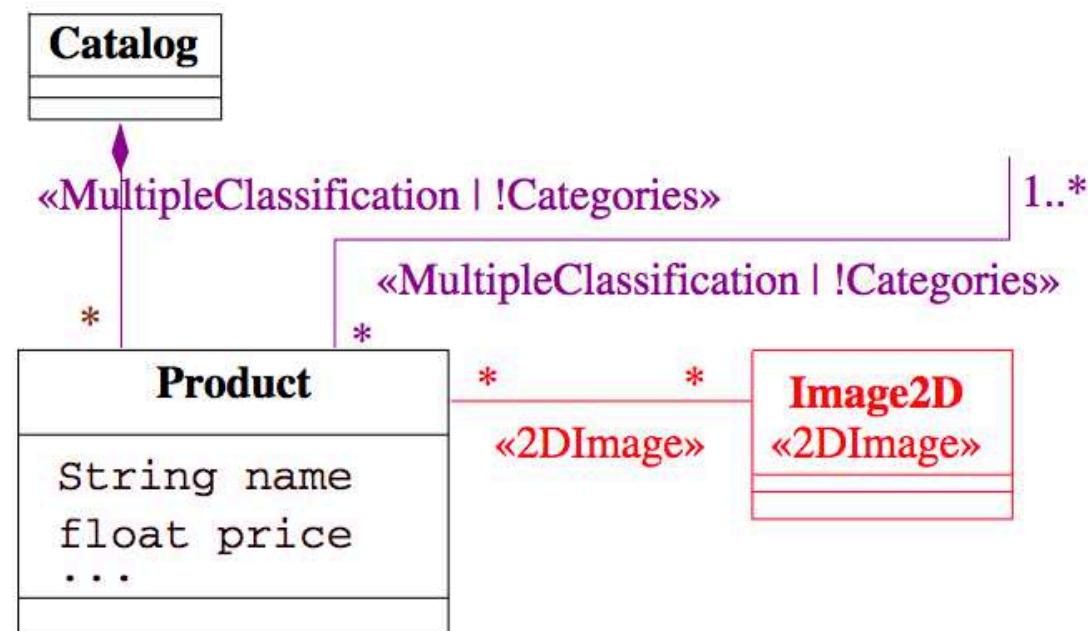
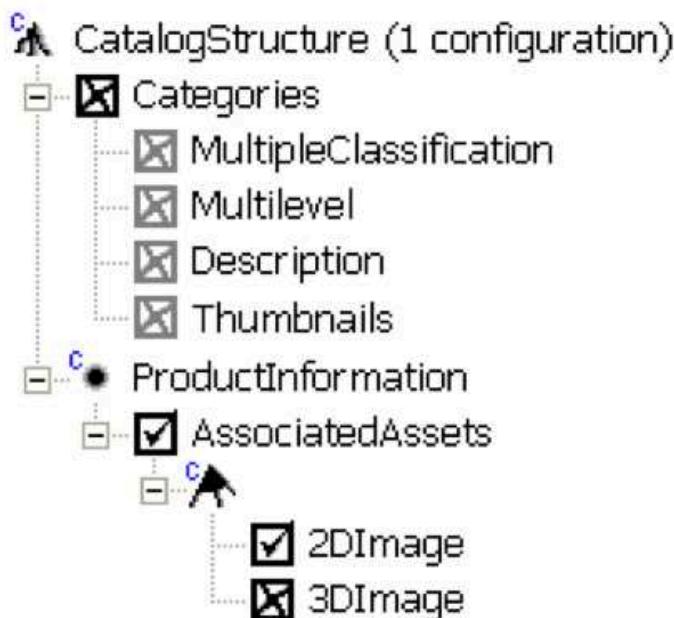
▲ CatalogStructure (52 configurations)

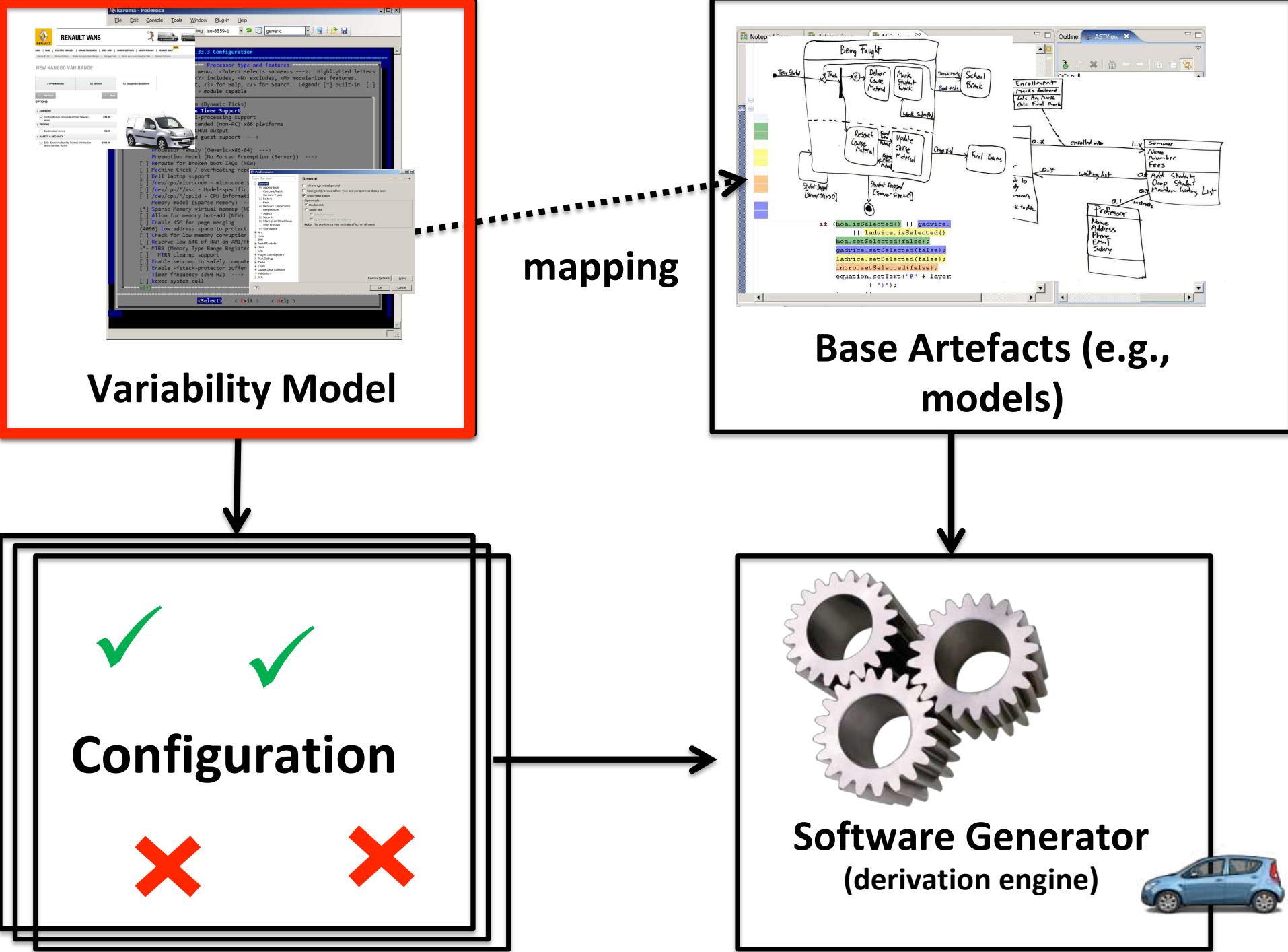
- ● Categories
 - MultipleClassification
 - Multilevel
 - Description
 - Thumbnails
- ● ProductInformation
 - AssociatedAssets
 - 2DImage
 - 3DImage





Ooops





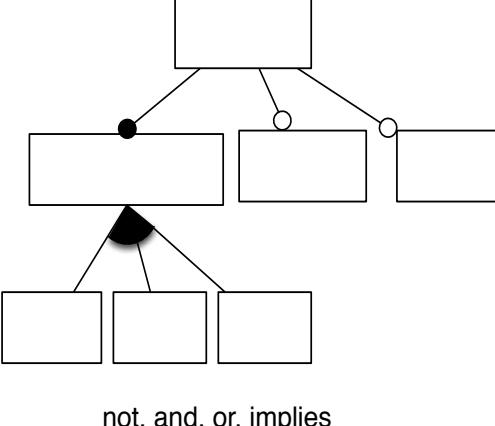
A photograph of an old, rusted pickup truck. The truck is heavily damaged, with its front end crushed and the driver's side door missing. It is situated in a field of tall, dry grass and weeds, with a dense thicket of bushes in the background. The overall scene conveys a sense of abandonment and decay.

Unused flexibility

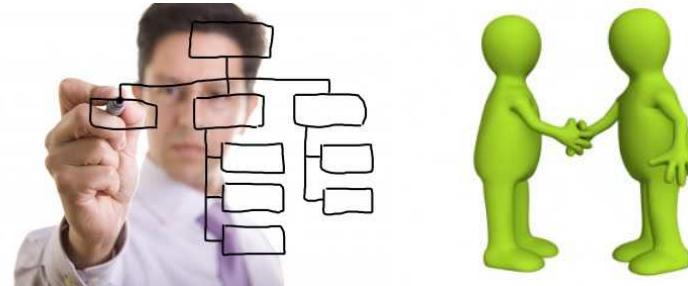


Illegal variant

Feature Model



Communicative



Analytic



Generative



Feature Models

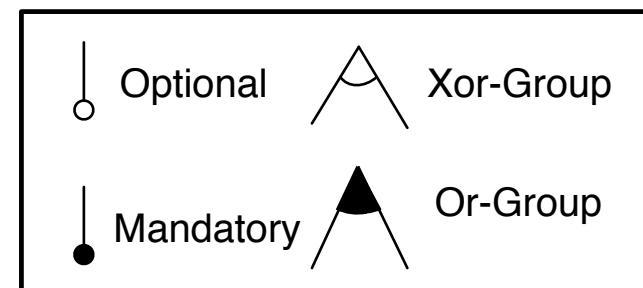
(defacto standard for modeling variability)

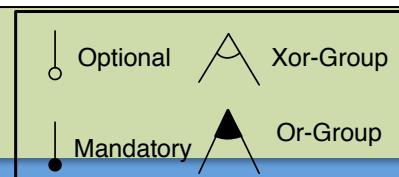
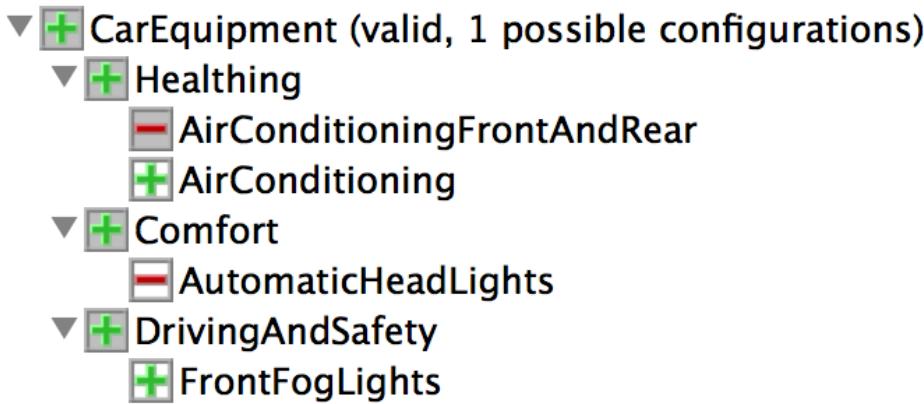


Hierarchy: rooted tree

Variability:

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





Hierarchy + Variability

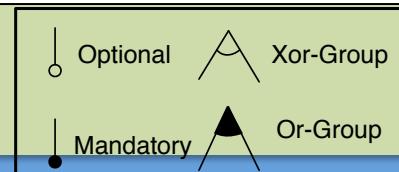
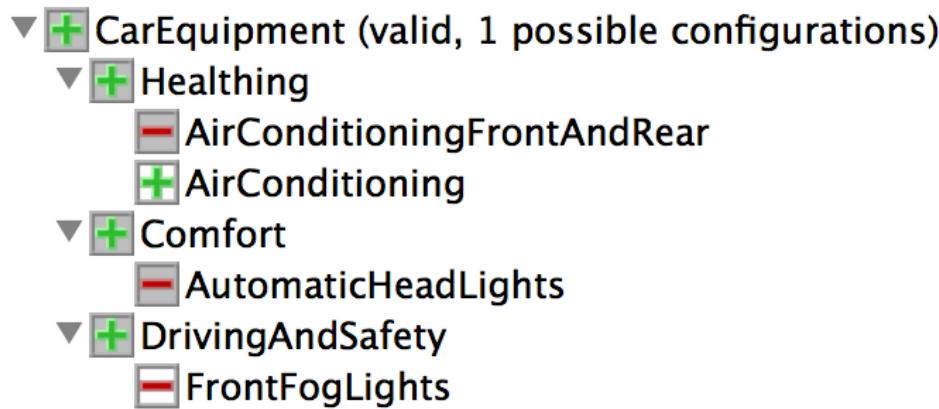
=

set of valid configurations

configuration = set of features selected

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





Hierarchy + Variability

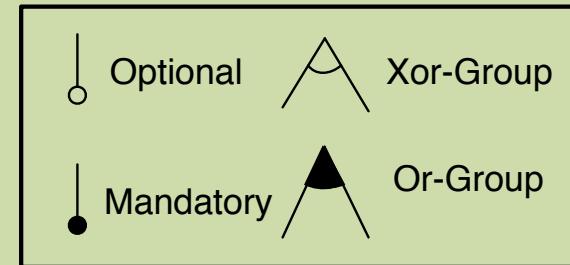
=

set of valid configurations

configuration = set of features selected

{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning}





Hierarchy + Variability

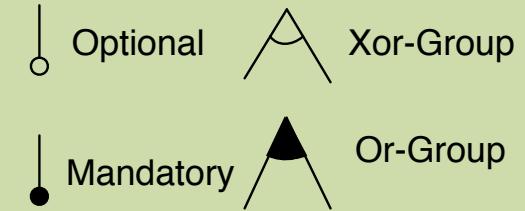
=

set of valid configurations

configuration = set of features selected

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



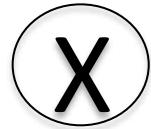


Hierarchy + Variability

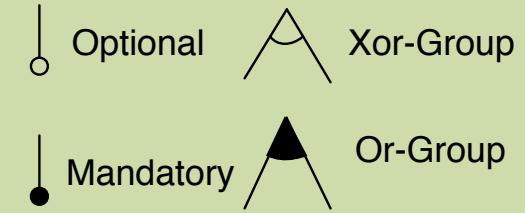
=

set of valid configurations

{CarEquipment, Comfort,
DrivingAndSafety,
Healthing}



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



Hierarchy + Variability

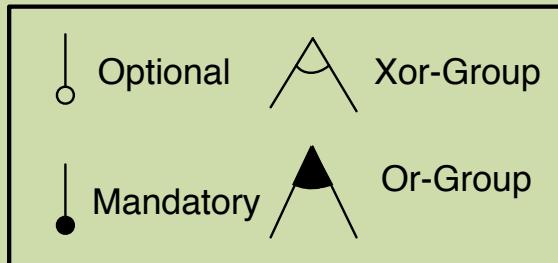
=

set of valid configurations

Configuration set (from a basic feature model of car)

	CarEquipment	Comfort	DrivingAndSafety	Healthing	AirConditioning	FrontFogLights	AutomaticHeadLights	AirConditioningFrontAndRear
Car2	yes	yes	yes	yes	yes	yes	yes	no
Car6	yes	yes	yes	yes	no	yes	no	yes
Car1	yes	yes	yes	yes	yes	yes	no	no
Car4	yes	yes	yes	yes	no	no	no	yes
Car5	yes	yes	yes	yes	yes	no	no	no
Car3	yes	yes	yes	yes	no	yes	yes	yes

ar}



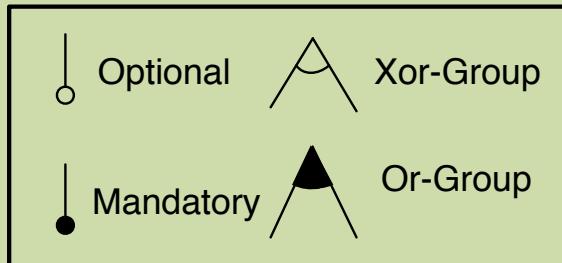
Hierarchy + Variability

=

set of valid configurations



Product ▲ ▼	CarEquipment ▼	Comfort ▼	DrivingAndSafety ▼	Healthing ▼	AirConditioning ▼	FrontFogLights ▼	AutomaticHeadLights ▼	AirConditioningFrontAndRear ▼
	Find	Yes <input type="checkbox"/> No <input type="checkbox"/>						
Car1	yes	yes	yes	yes	yes	yes	no	no
Car2	yes	yes	yes	yes	yes	yes	yes	no
Car3	yes	yes	yes	yes	no	yes	yes	yes
Car4	yes	yes	yes	yes	no	no	no	yes
Car5	yes	yes	yes	yes	yes	no	no	no
Car6	yes	yes	yes	yes	no	yes	no	yes



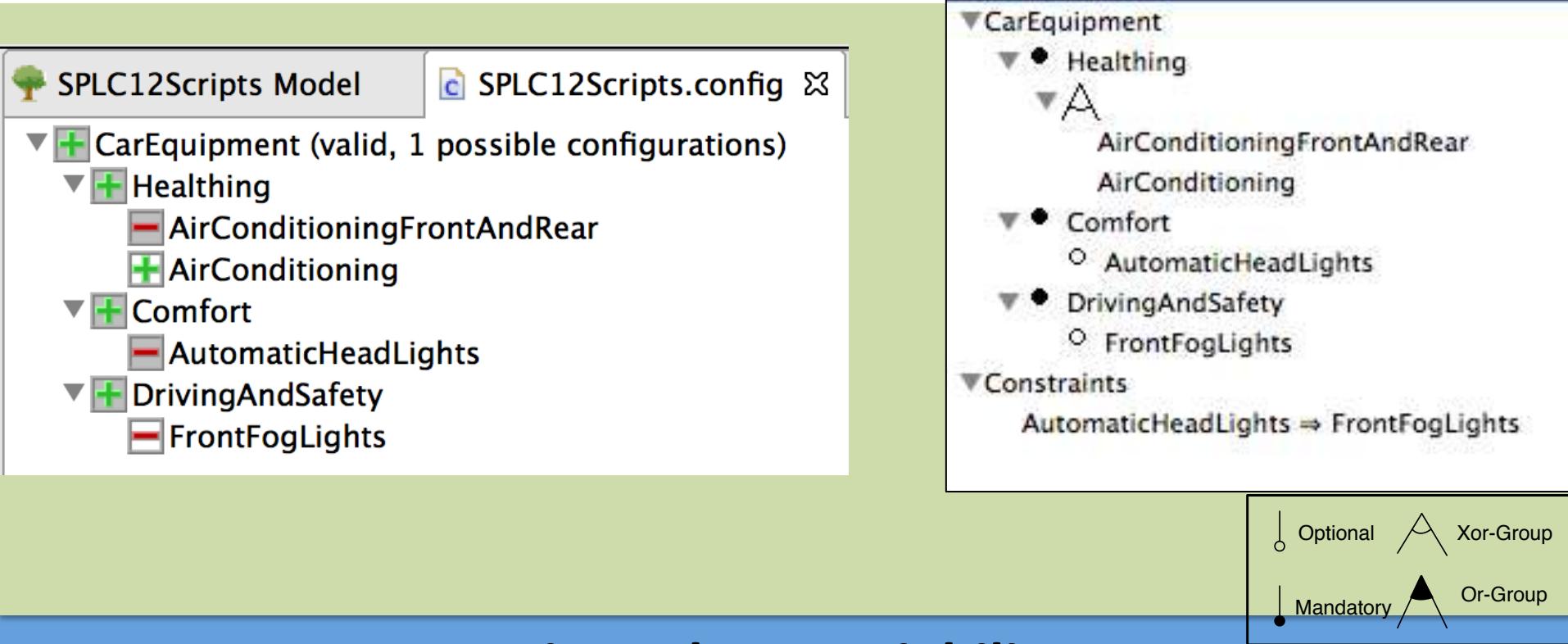
Hierarchy + Variability

=

set of valid configurations



Product ▾ ▾	▼	▼	▼	▼	▼	AirConditioning ▾	FrontFogLights ▾	AutomaticHeadLights ▾	AirConditioningFrontAndRear ▾
Find						Yes <input type="checkbox"/> No <input type="checkbox"/>			
Car1						yes	yes	no	no
Car2						yes	yes	yes	no
Car3						no	yes	yes	yes
Car4						no	no	no	yes
Car5						yes	no	no	no
Car6						no	yes	no	yes



Hierarchy + Variability

=

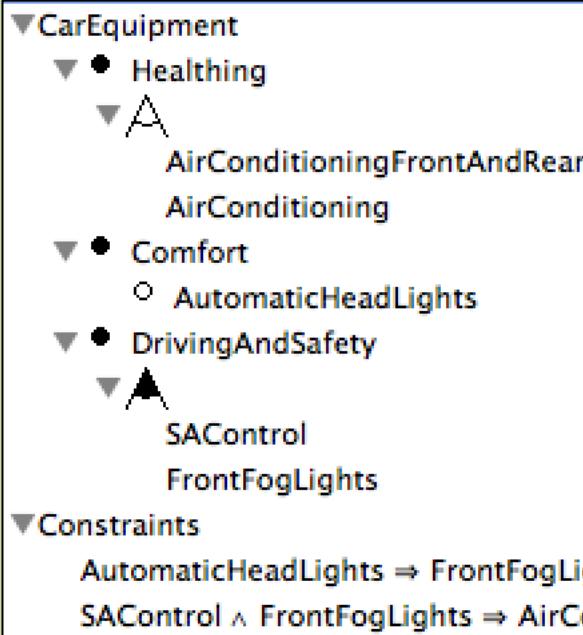
set of valid configurations

configuration = set of features selected

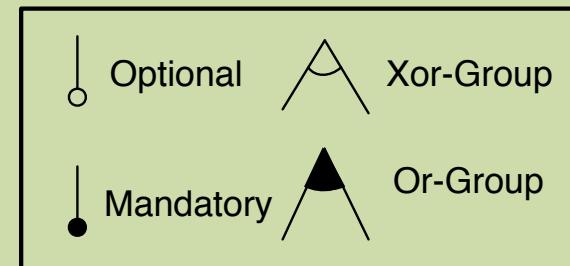
{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning}

Product						AirConditioning		FrontFogLights		AutomaticHeadLights		AirConditioningFrontAndRear
	Find					Yes <input checked="" type="checkbox"/> No <input type="checkbox"/>		Yes <input type="checkbox"/> No <input checked="" type="checkbox"/>		Yes <input type="checkbox"/> No <input checked="" type="checkbox"/>		Yes <input type="checkbox"/> No <input checked="" type="checkbox"/>
Car5						yes		no		no		no





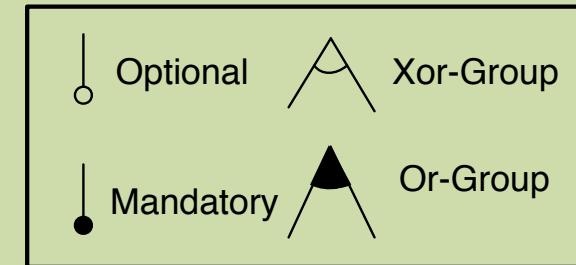
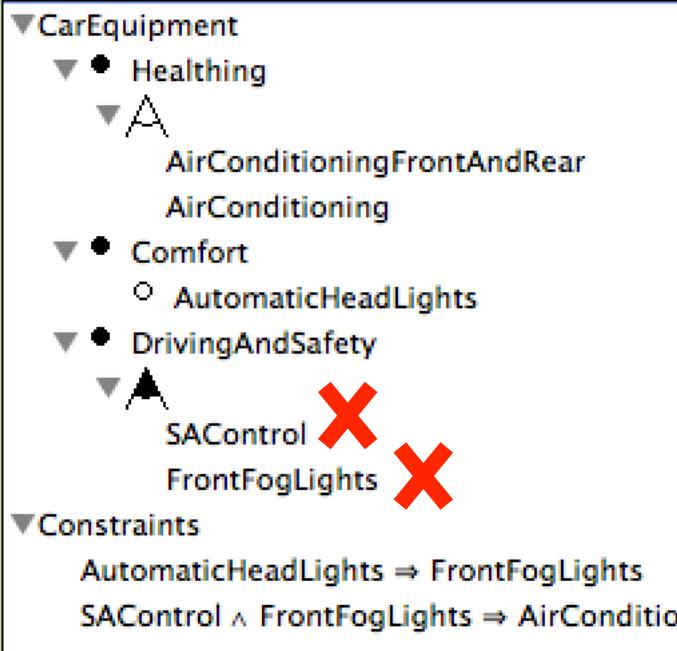
Boolean logic: \wedge , \vee , not, implies



Hierarchy + Variability
=

set of valid configurations





Hierarchy + Variability

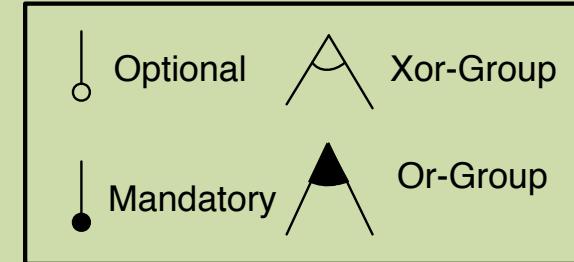
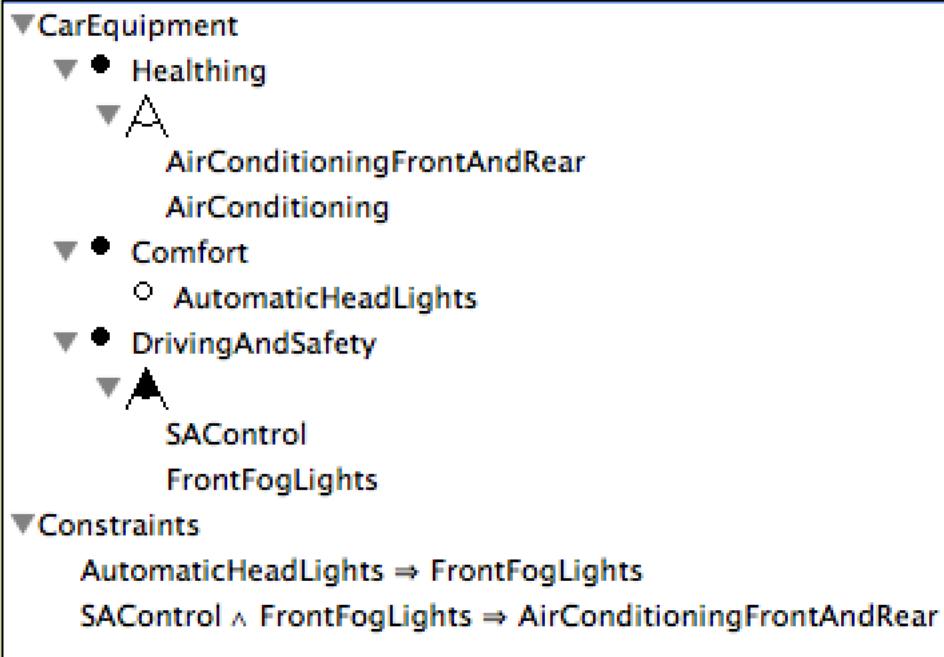
=

set of valid configurations



Or-group: at least one!





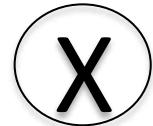
Hierarchy + Variability

=

set of valid configurations



{CarEquipment, Comfort,
DrivingAndSafety,
Healthing}



- {AirConditioningFrontAndRear, FrontFogLights, SAControl}
- {AirConditioningFrontAndRear, SAControl}
- {AutomaticHeadLights, AirConditioning, FrontFogLights}
- {AirConditioningFrontAndRear, SAControl, AutomaticHeadLights, FrontFogLights}
- {FrontFogLights, AirConditioning}
- {AutomaticHeadLights, AirConditioningFrontAndRear, FrontFogLights}
- {FrontFogLights, AirConditioningFrontAndRear}
- {SAControl, AirConditioning}



(Boolean)
Feature Models



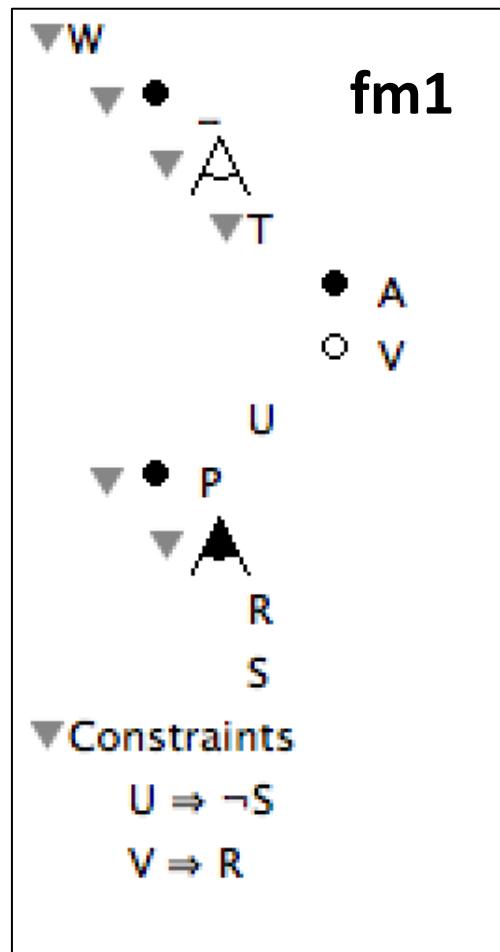
(Boolean)
Formula φ

Product ▾ ▾	CarEquipment ▾	Comfort ▾	DrivingAndSafety ▾	Healting ▾	AirConditioning ▾	FrontFogLights ▾	AutomaticHeadLights ▾	AirConditioningFrontAndRear ▾
Find	Yes <input type="checkbox"/> No <input type="checkbox"/>							
Car1	yes	yes	yes	yes	yes	yes	no	no
Car2	yes	no						
Car3	yes	yes	yes	yes	no	yes	yes	yes
Car4	yes	yes	yes	yes	no	no	no	yes
Car5	yes	yes	yes	yes	yes	no	no	no
Car6	yes	yes	yes	yes	no	yes	no	yes

(Boolean)
Product Comparison Matrix

(Boolean) Feature Models

Hierarchy + Variability = set of valid configurations



$\llbracket fm1 \rrbracket = \{$

$\{W, P, R, S, T, A, V\},$

$\{W, P, S, T, A\},$

$\{W, P, R, T, A\},$

$\{W, P, R, U\},$

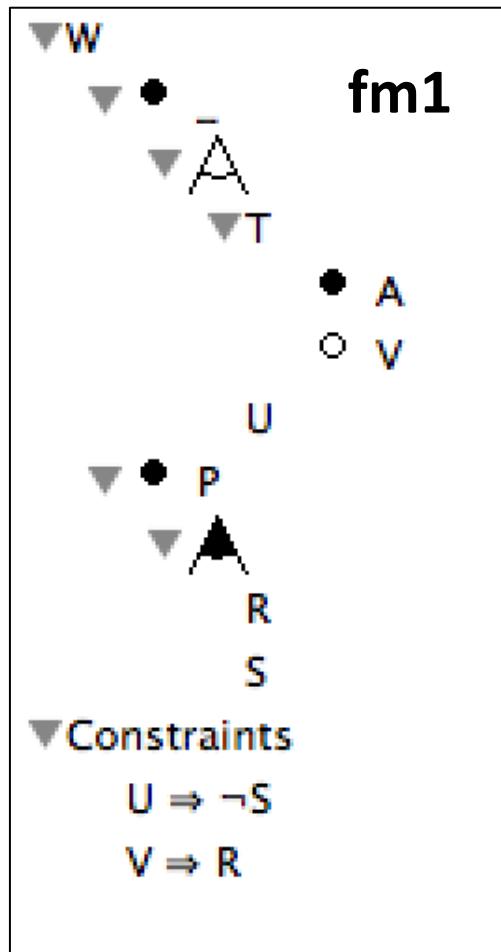
$\{W, P, R, T, V, A\},$

$\{W, P, R, S, T, A\},$

$\}$

(Boolean) Feature Models

~ Boolean formula

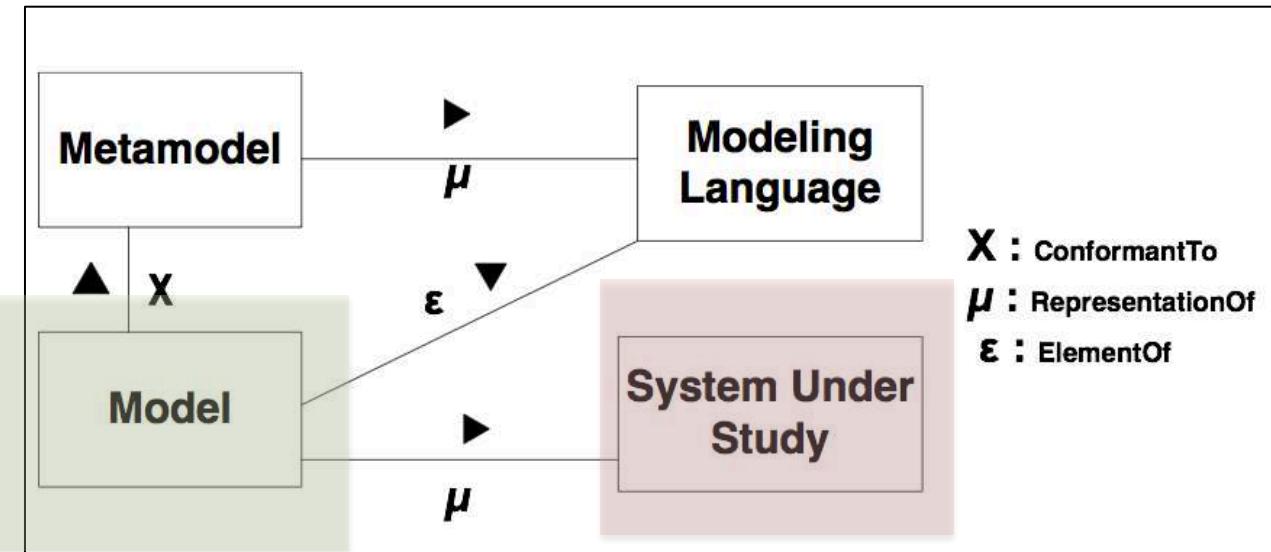


$\phi_{fm_1} = W // \text{root}$
 $\wedge W \Leftrightarrow P // \text{mandatory}$
 $// \text{Or-group}$
 $\wedge P \Rightarrow R \vee S$
 $\wedge R \Rightarrow P \wedge S \Rightarrow P$
 $\wedge V \Rightarrow T // \text{optional}$
 $\wedge A \Leftrightarrow T // \text{mandatory}$
 $// \text{Xor-group}$
 $\wedge T \Rightarrow W$
 $\wedge U \Rightarrow W$
 $\wedge \neg T \vee \neg U$
 $// \text{constraints}$
 $\wedge V \Rightarrow R // \text{implies}$
 $\wedge \neg U \Rightarrow \neg S // \text{excludes}$

Languages:

How to specify feature models?

Feature Models



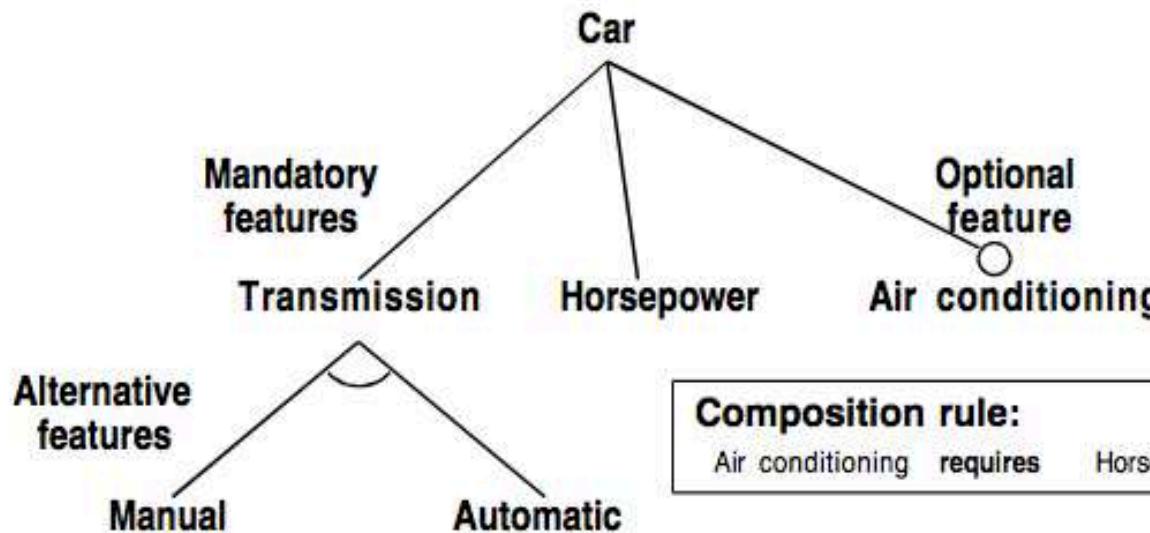
History of Feature Models

~ often called **Feature Diagrams**
(suggest a visual representation)

graphical languages

Feature Models

Kang et al. (1990)

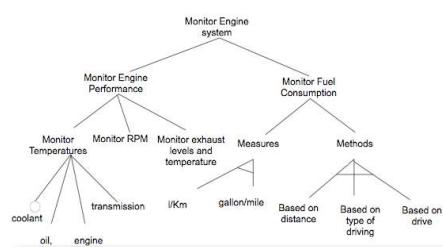


Composition rule:

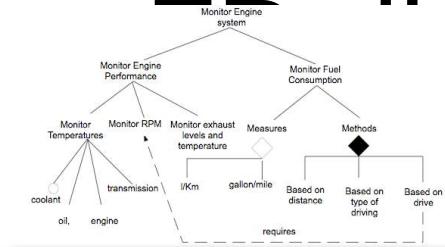
Air conditioning requires Horsepower > 100

Rationale:

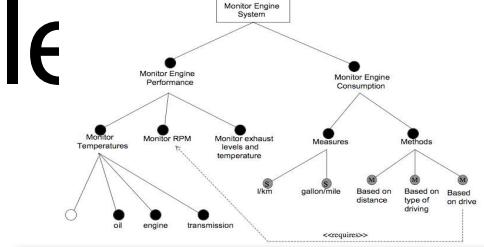
Manual more fuel efficient



FODA (OFT)
[Kang et al., 1990]



FeatuRSEB (RFD)
[Griss et al., 1998]



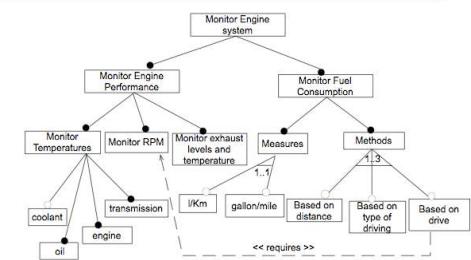
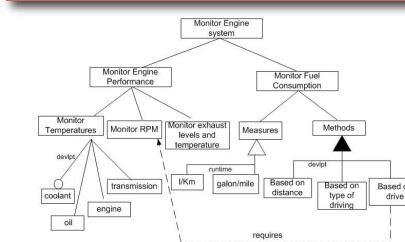
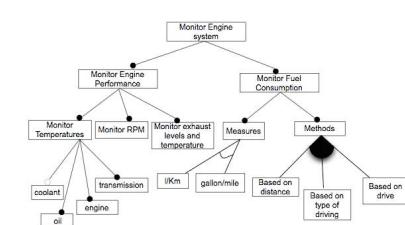
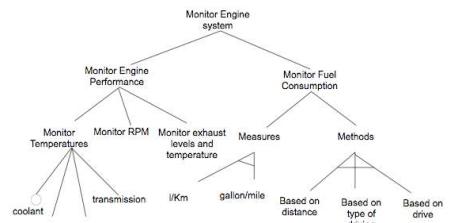
PLUSS (PFT)
[Eriksson et al., 2005]

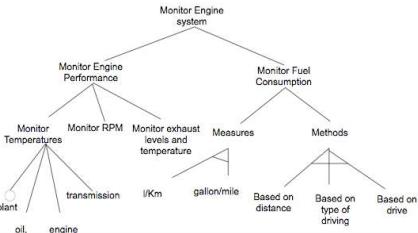
FORM (OFD)
[Kang et al., 1998]

Gen. Prog. (GPFT)
[Czarnecki et al., 2000]

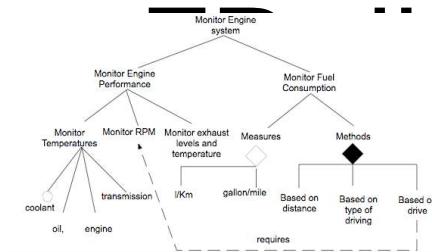
VBFD
[van Gurp et al., 2001]

EFD
[Riebisch et al., 2002]

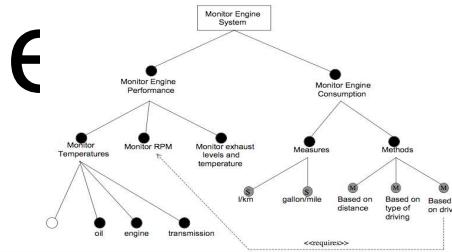




FODA (OFT)
[Kang et al., 1990]



FeatuRSEB (RFD)
[Griss et al., 1998]

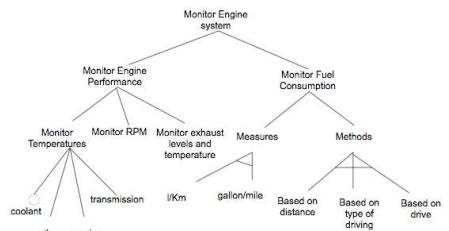


PLUSS (PFT)
[Eriksson et al., 2005]

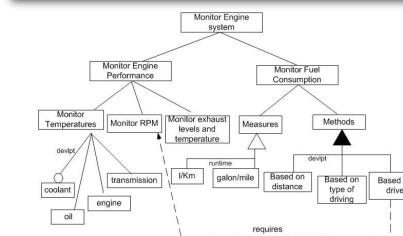
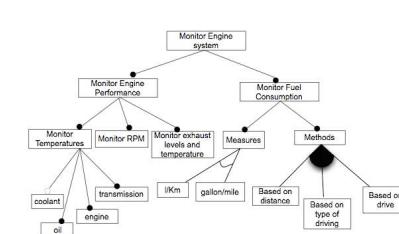
Aesthetic differences

FORM (OFD)
[Kang et al., 1998]

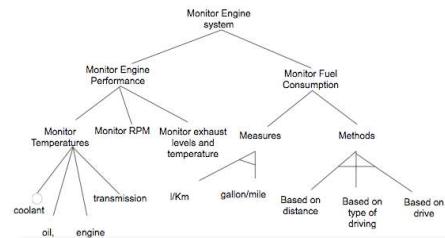
Gen. Prog. (GPFT)
[Czarnecki et al., 2000]



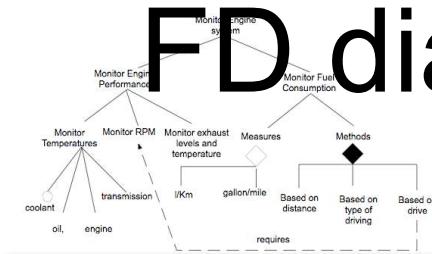
VBFD
[van Gurp et al., 2001]



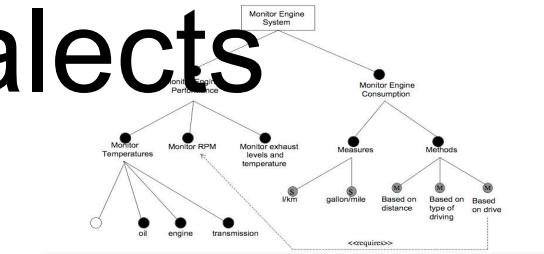
EFD
[Riebisch et al., 2002]



FODA (OFT)
[Kang et al., 1990]



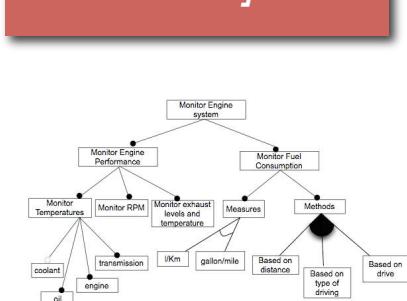
FeatuRSEB (RFD) [Griss et al., 1998]



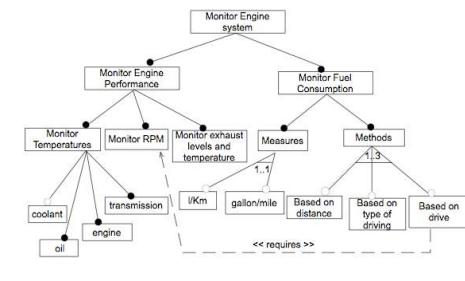
PLUSS (PFT)
[Eriksson et al., 2005]

Aesthetic differences

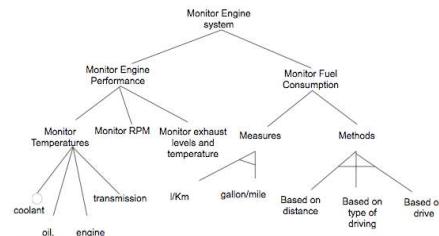
FORM (OFD) [Kang et al., 1998]



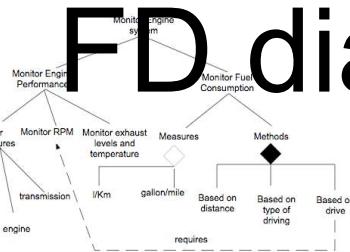
VBFD
[van Gurp et al.,
2001]



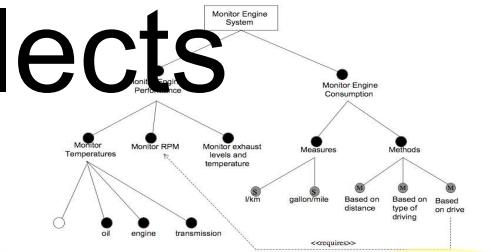
FD dialects



FODA (OFT)
[Kang et al., 1990]



FeatuRSEB (RFD)
[Griss et al., 1998]



PLUSS (PFT)
[Eriksson et al., 2001]

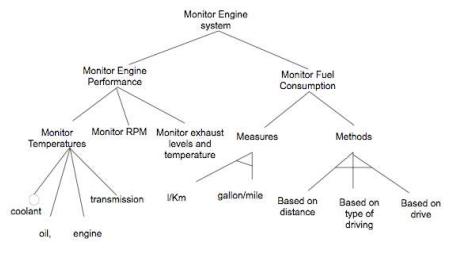
Aesthetic differences

Stronger claims

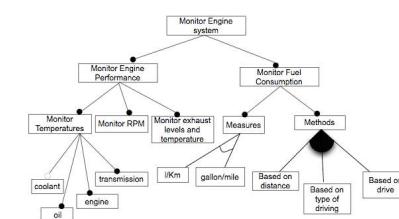
Vague use of terms
syntax, semantics,
expressiveness, ambiguity ...

FORM (OFD)
[Kang et al., 1998]

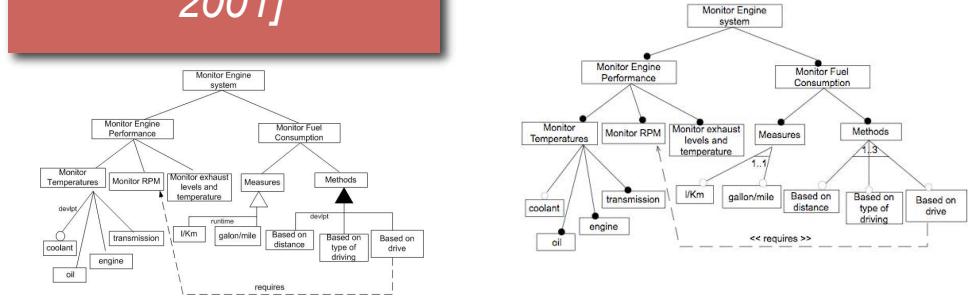
Gen. Prog. (GPFT)
[Czarnecki et al., 2000]



VBFD
[van Gurp et al., 2001]



EFD
[Riebisch et al., 2002]



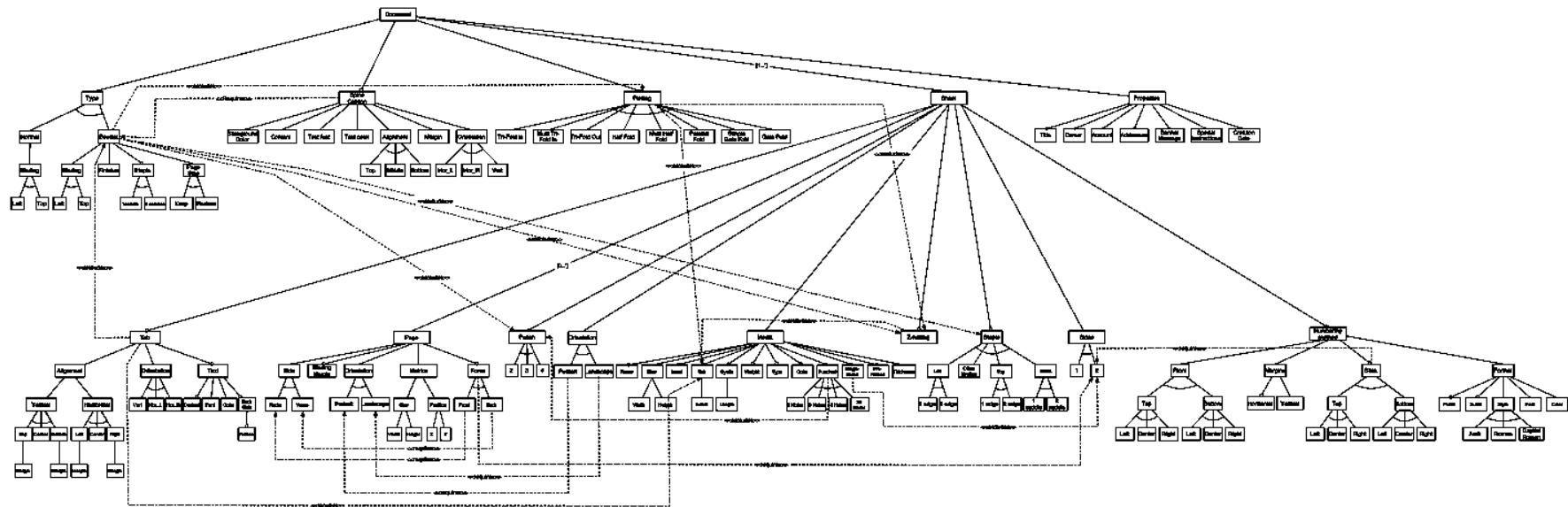
Concrete (graphical)
syntax matters...

But semantics even more!

Feature Models

~ Feature Diagrams
(suggest a visual representation)

but that's not mandatory
that does not necessary scale
or respond to users' concerns



Textual variability languages

- Andreas Classen, Quentin Boucher, Patrick Heymans: A text-based approach to feature modelling: Syntax and semantics of TVL. *Sci. Comput. Program.* 76(12): 1130-1143 (2011)
- Czarnecki et al. “Clafer: Unifying Class and Feature Modeling” SoSyM’15
- Mathieu Acher, Philippe Collet, Philippe Lahire, Robert B. France « A Domain-Specific Language for Large-Scale Management of Feature Models » *Science of Computer Programming* (SCP), 2013
- Mauricio Alférez, José A. Galindo, Mathieu Acher, and Benoit Baudry. Modeling Variability in the Video Domain: Language and Experience Report (2014). Research Report RR-8576

FAMILIAR

(FeAture Model script Language for manipulation and Automatic Reasoning)

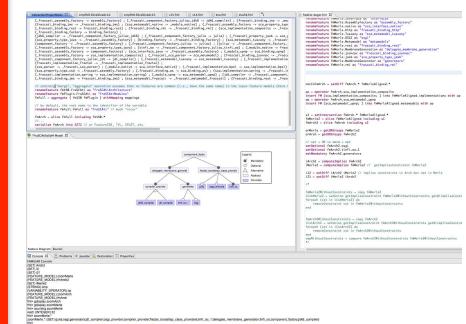
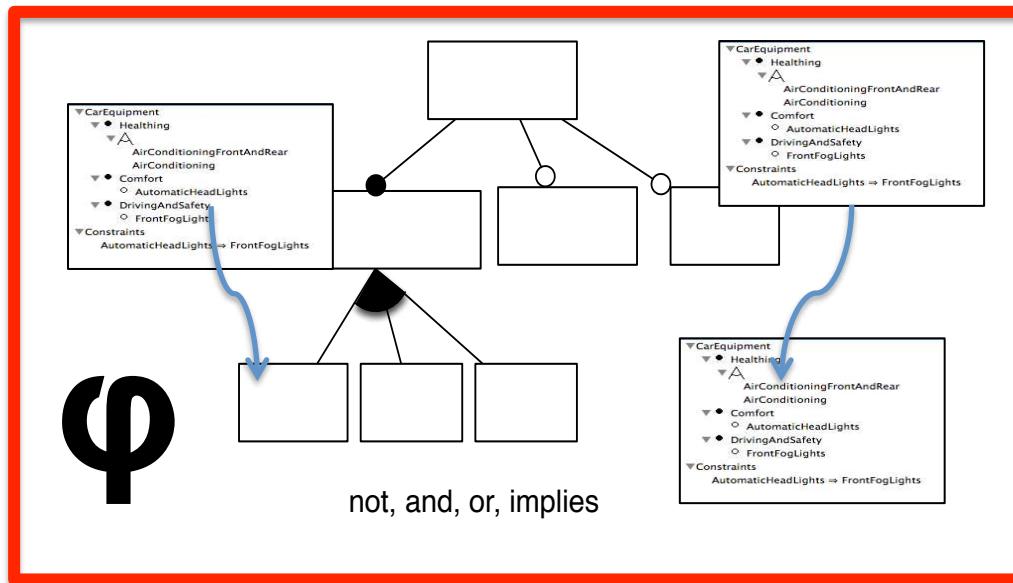
<http://familiar-project.github.com/>

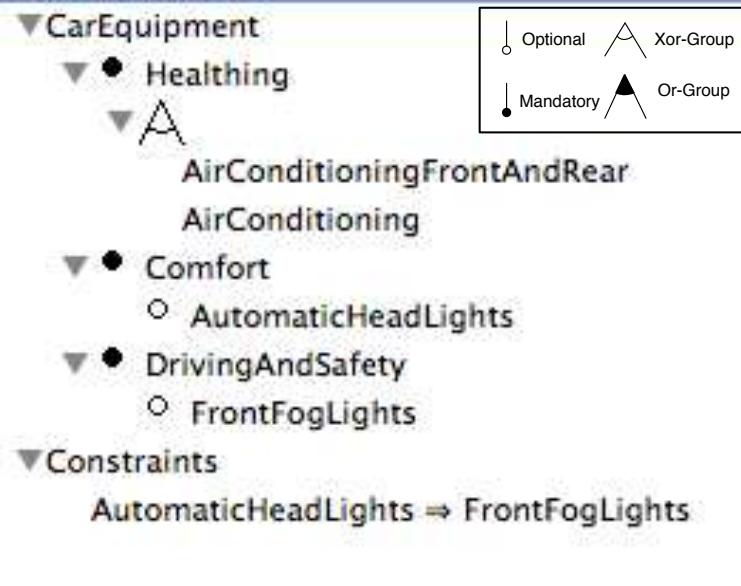


S.P.L.O.T.
Software Product Lines Online Tools

IDE
Feature

TVL
DIMACS



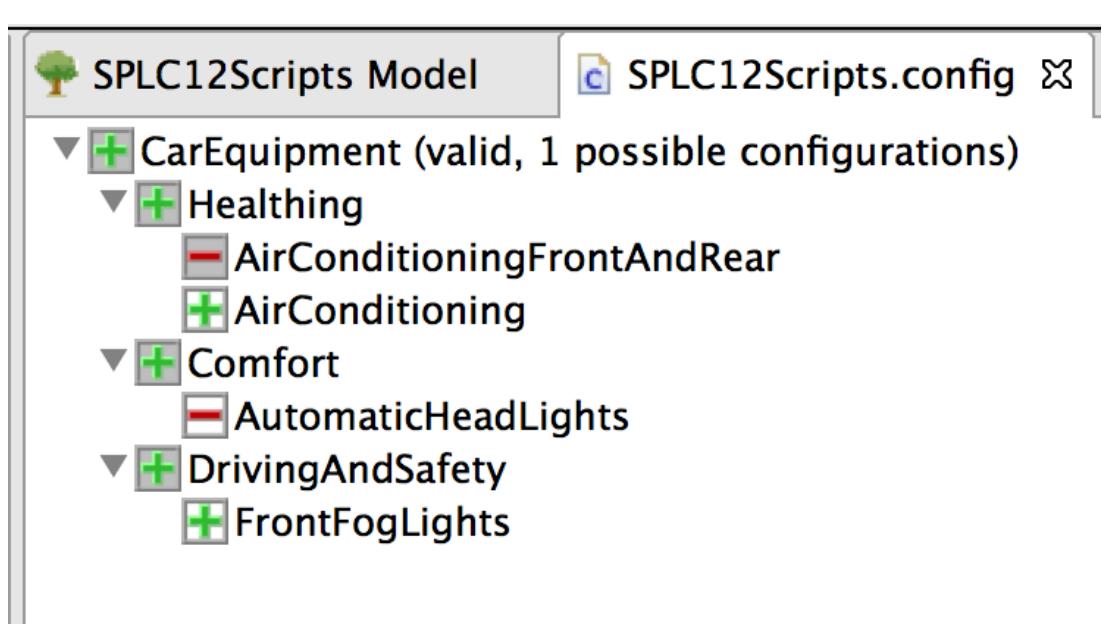
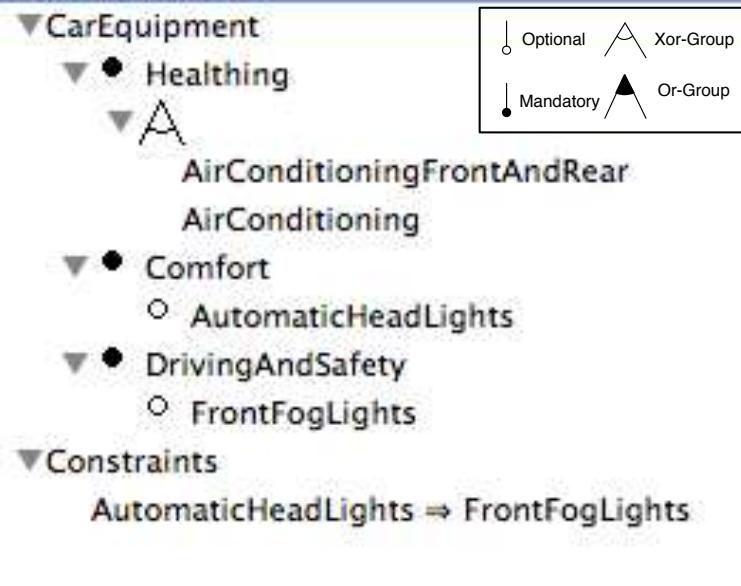


```
fml> convert fmCarEquipment into SPLIT
res1: (STRING) <feature_model name="fmCarEquipment">
<meta>
<data name="description"/>
<data name="creator"/>
<data name="address"/>
<data name="email"/>
<data name="phone"/>
<data name="website"/>
<data name="organization"/>
<data name="department"/>
<data name="date"/>
<data name="reference"/>
</meta>
<feature_tree>
: r CarEquipment(_r0)
  : m Healthing(_r1)
    : g [1,1]
      : AirConditioningFrontAndRear(_r2)
      : AirConditioning(_r3)
    : m DrivingAndSafety(_r4)
      : o FrontFogLights(_r5)
    : m Comfort(_r6)
      : o AutomaticHeadLights(_r7)
</feature_tree>
<constraints>
C0: ~_r7 or _r5
</constraints>
</feature_model>
```

S.P.L.O.T.
Software Product Lines Online Tools
IDE
eature

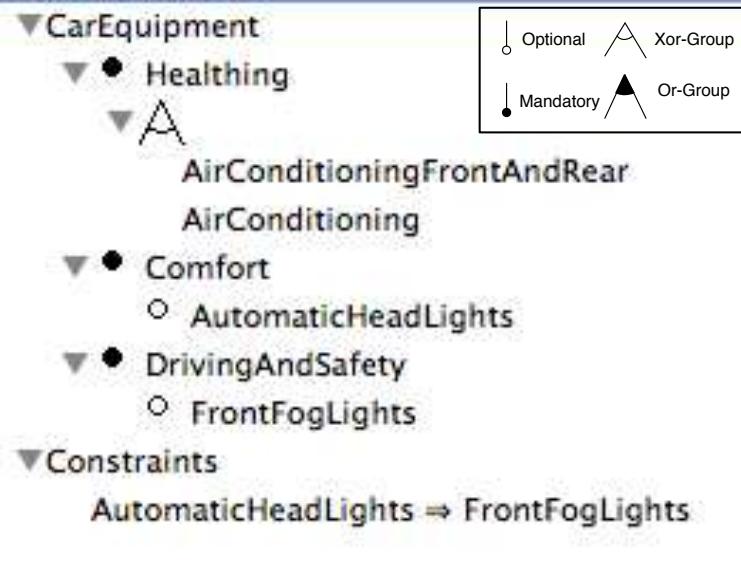
```
fmCarEquipment = FM (CarEquipment : Healthing DrivingAndSafety Comfort ; // 3 mandatory features
                      Healthing : (AirConditioning|AirConditioningFrontAndRear) ; // xor
                      DrivingAndSafety : [FrontFogLights] ; // optional
                      Comfort : [AutomaticHeadLights]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

```
MacBook-Pro-de-Mathieu-3:Documents mache1$ java -Xmx1024M -jar FML-basic-1.1.jar FMLTestRepository/carEquipTuto.fml
FAMILIAR (for Feature Model script Language for manipulation and Automatic Reasoning) version 1.1 (beta)
http://familiar-project.github.com/
fml> ls
(FEATURE_MODEL) fmCarEquipment
fml>
```

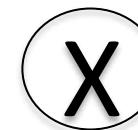


```
fmlCarEquipment = FM (CarEquipment : Healthing DrivingAndSafety Comfort ; // 3 mandatory features
                      Healthing : (AirConditioning|AirConditioningFrontAndRear) ; // Xor
                      DrivingAndSafety : [FrontFogLights] ; // optional
                      Comfort : [AutomaticHeadLights]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

```
fml> c1 = configuration fmlCarEquipment
c1: (CONFIGURATION) selected: [Healthing, CarEquipment, DrivingAndSafety, Comfort]           deselected: []
fml> select AirConditioning FrontFogLights in c1
res2: (BOOLEAN) true
fml> deselect AutomaticHeadLights in c1
res3: (BOOLEAN) true
fml> selectedF c1
res4: (SET) {Comfort;CarEquipment;Healthing;AirConditioning;DrivingAndSafety;FrontFogLights}
```



{CarEquipment, Comfort,
DrivingAndSafety,
Healthing}



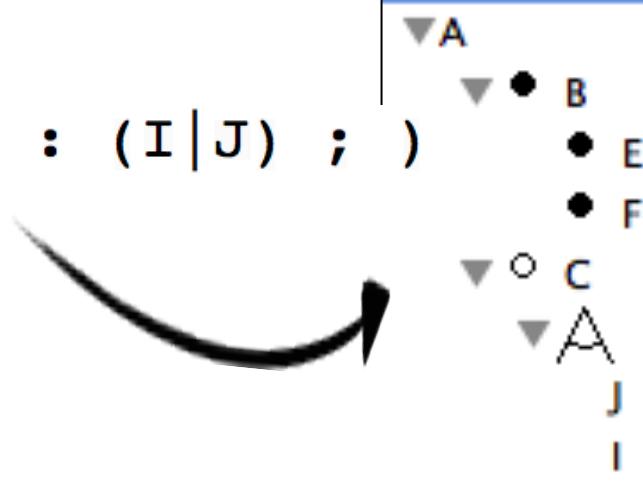
{AirConditioning, FrontFogLights}
 {AutomaticHeadLights, AirConditioning,
 FrontFogLights}
 {AutomaticHeadLights, FrontFogLights,
 AirConditioningFrontAndRear}
 {AirConditioningFrontAndRear}
 {AirConditioning}
 {AirConditioningFrontAndRear, FrontFogLights}

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

```
fml> s1 = configs fmlCarEquipment
s1: (SET) {{Comfort;CarEquipment;FrontFogLights;DrivingAndSafety;AirConditioning;Healthing};{AirConditioningFrontAndRear;CarEquipment;Comfort;DrivingAndSafety;Healthing};{FrontFogLights;DrivingAndSafety;AutomaticHeadLights;CarEquipment;Healthing;AirConditioningFrontAndRear;Comfort};{Healthing;CarEquipment;AirConditioning;DrivingAndSafety;Comfort};{Healthing;CarEquipment;Comfort;FrontFogLights;DrivingAndSafety;AirConditioningFrontAndRear};{AutomaticHeadLights;DrivingAndSafety;CarEquipment;AirConditioning;Healthing;Comfort;FrontFogLights}}
fml> size s1
res0: (INTEGER) 6
fml> counting fmlCarEquipment
res1: (DOUBLE) 6.0
```

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

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



```
r1 = root fml
```

```
s = children r1
```

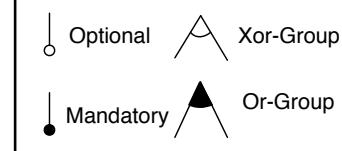
```
s1 = children fml.A
```

```
assert (s eq s1) // equality of the two sets
```

```
ft1 = parent fml.F
```

```
str1 = name ft1
```

```
ft2 = parent F // parent fml.F
```

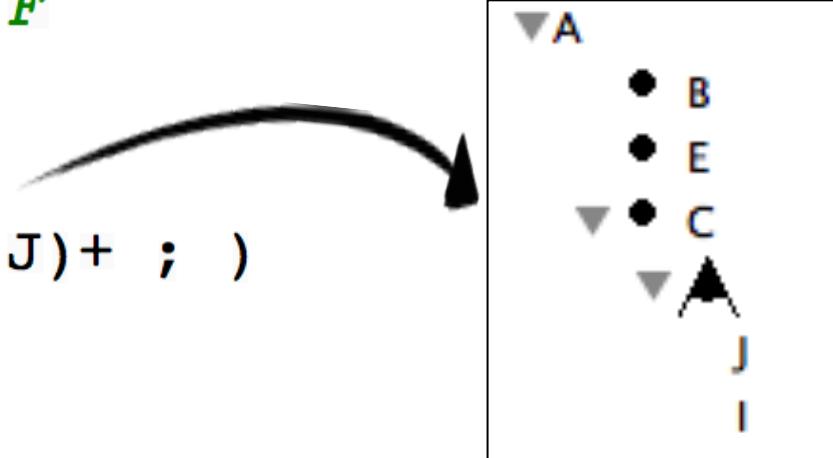


```
// another FM
```

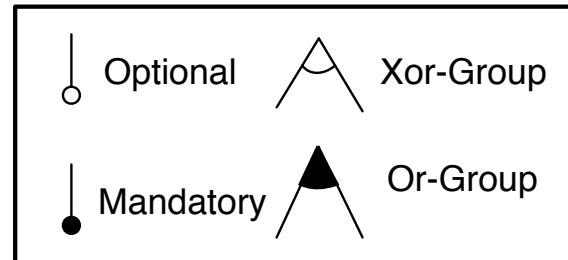
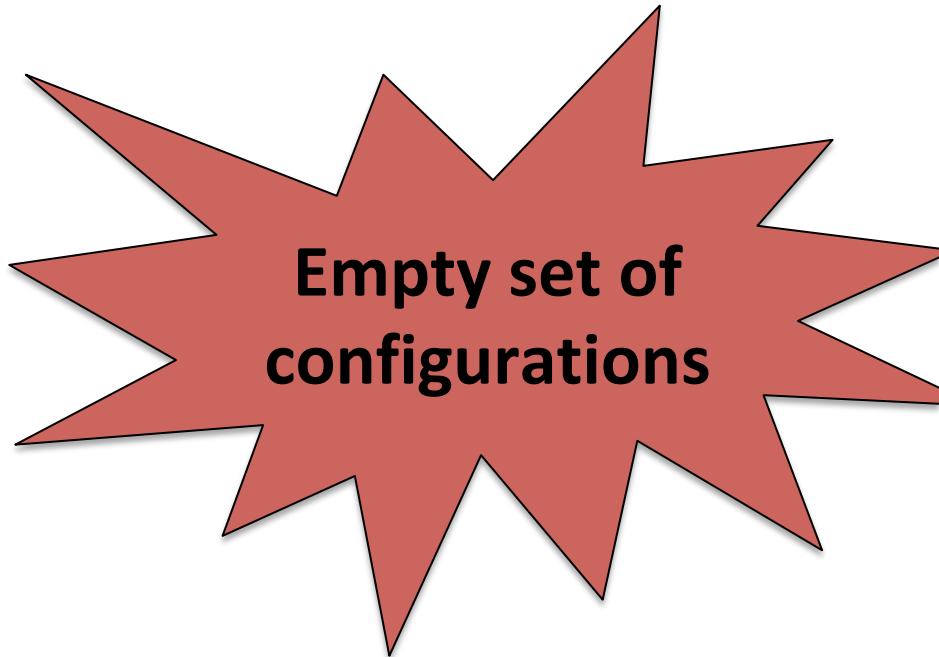
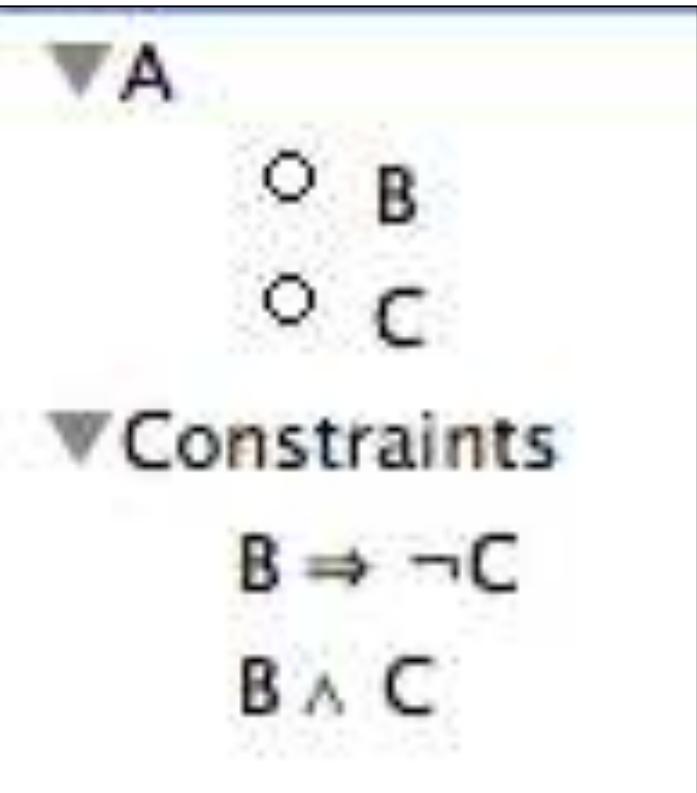
```
fm2 = FM (A : B C E ; C : (I|J)+ ; )
```

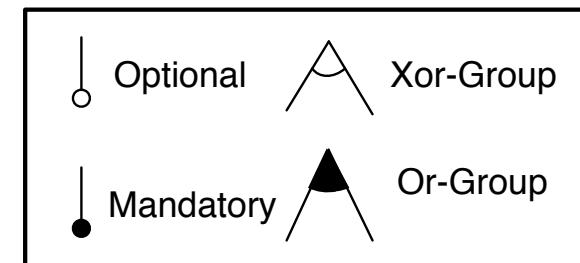
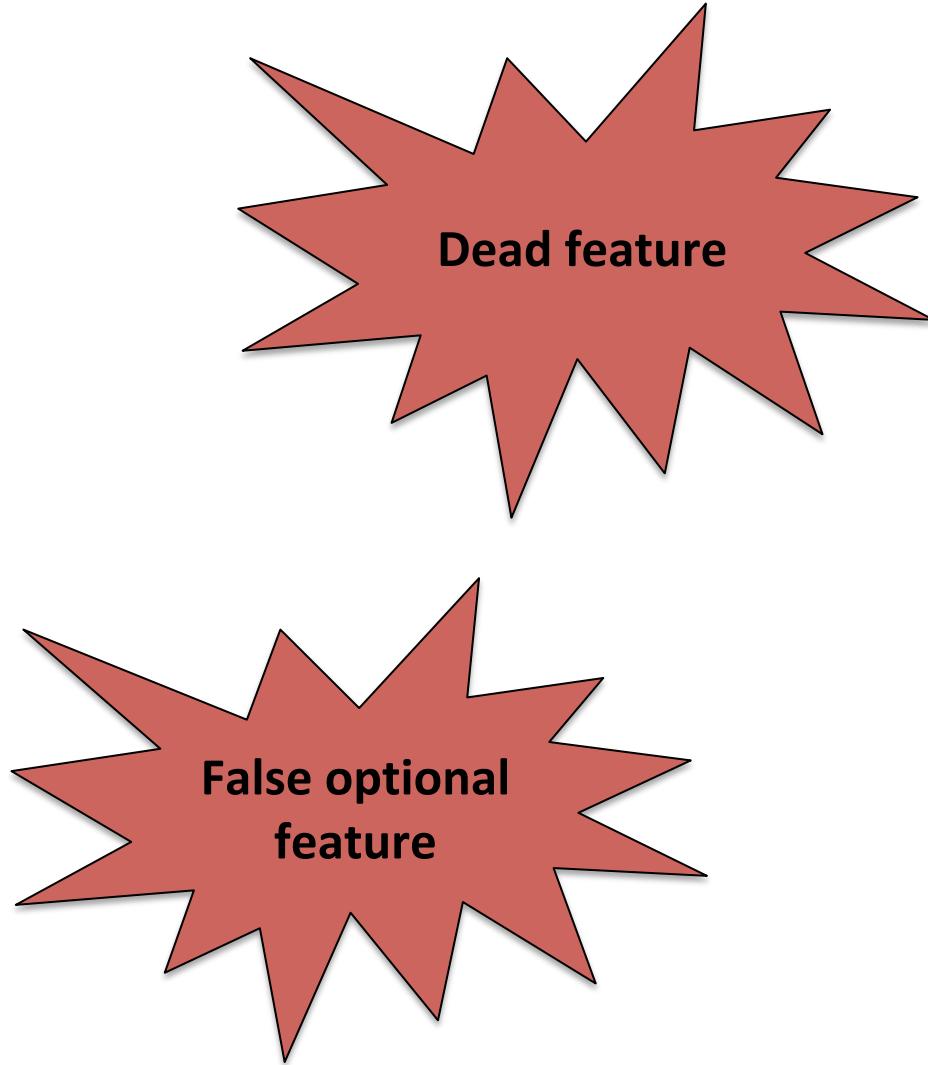
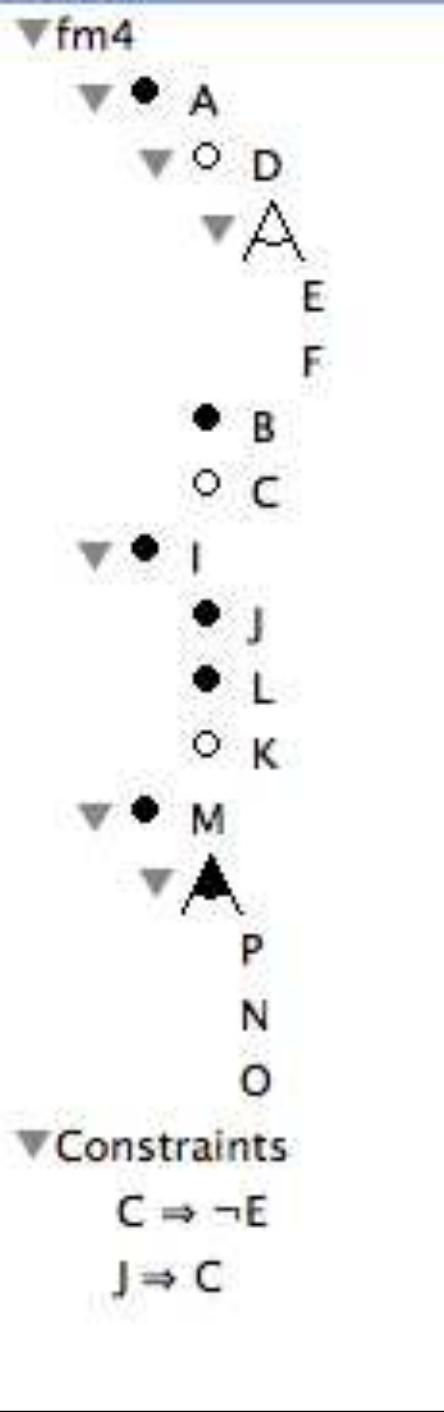
```
ft3 = fm2.B
```

```
ft4 = name B // ambiguity
```

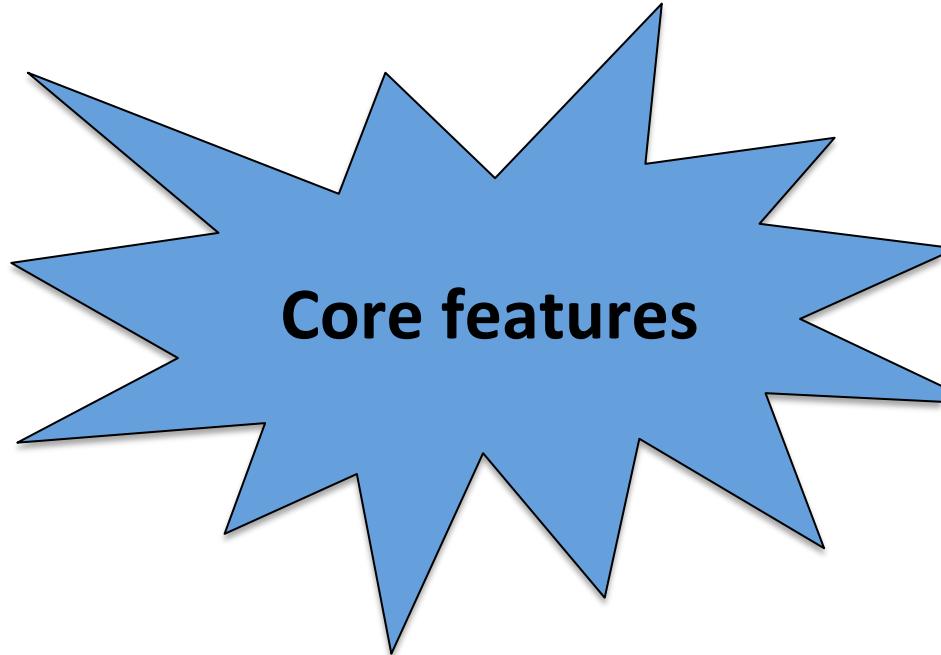


I want to analyze and
play with my specification!

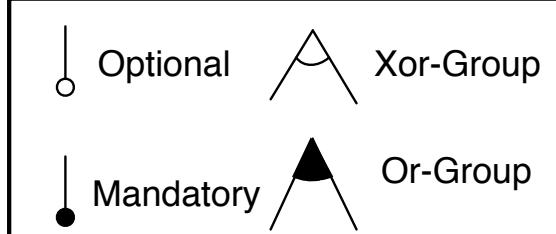


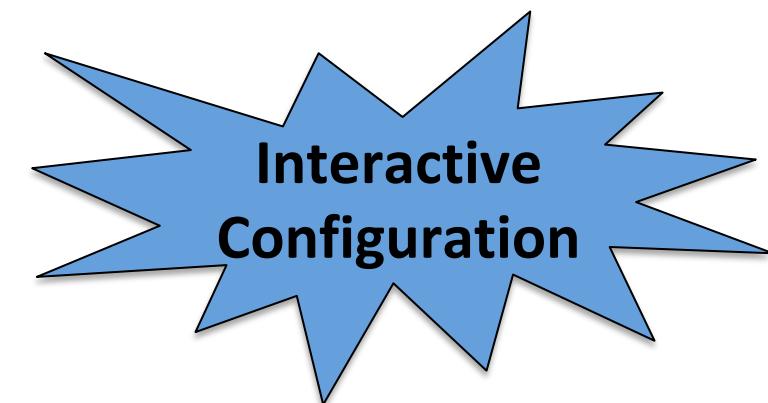
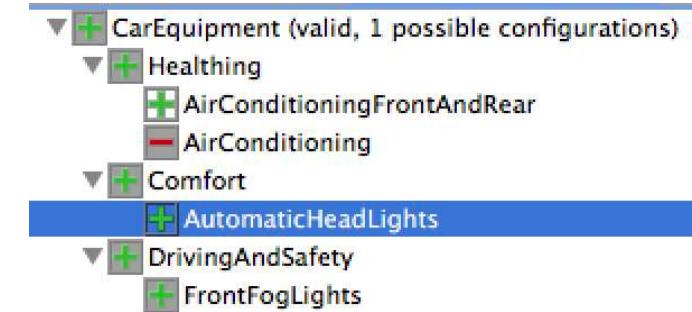
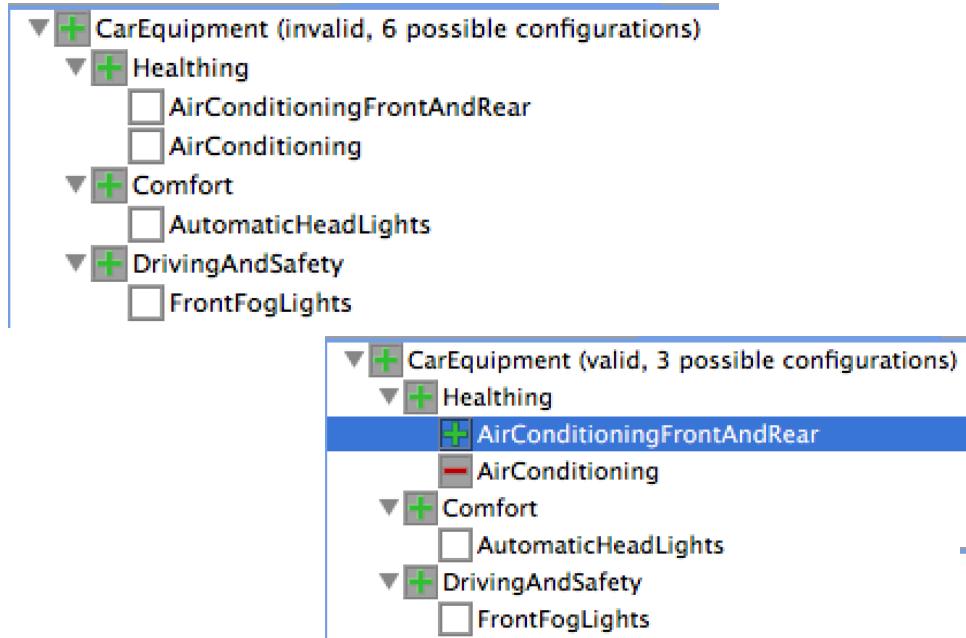
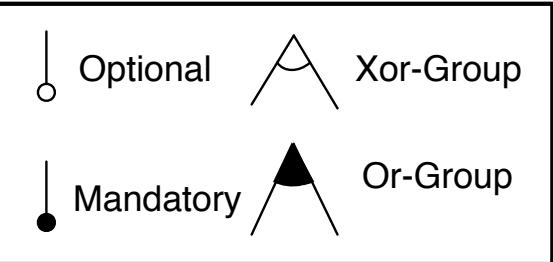


```
▼CarEquipment
  ▼● Healthing
  ▼△ A
    AirConditioningFrontAndRear
    AirConditioning
  ▼● Comfort
    ○ AutomaticHeadLights
  ▼● DrivingAndSafety
    ○ FrontFogLights
▼Constraints
  AutomaticHeadLights => FrontFogLights
```



{CarEquipment, Comfort,
DrivingAndSafety, Healthing}





Feature Models and Automated Reasoning

Benavides et al. survey, 2010

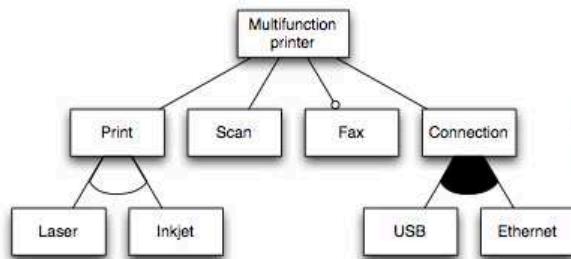
Decision problems and complexity

- Validity of a feature model
- Validity of a configuration
- Computation of dead and core features
- Counting of the number of valid configurations
- Equivalence between two feature models
- Satisfiability (SAT) problem
 - NP-complete

How to automate analysis of your feature models?

Binary Decision Diagram (BDD)
SAT solver

Typical implementations



result



logics

solvers



Z3

Feature Models

