Hack your DSL (Project)

Mathieu Acher
Maître de Conférences
mathieu.acher@irisa.fr

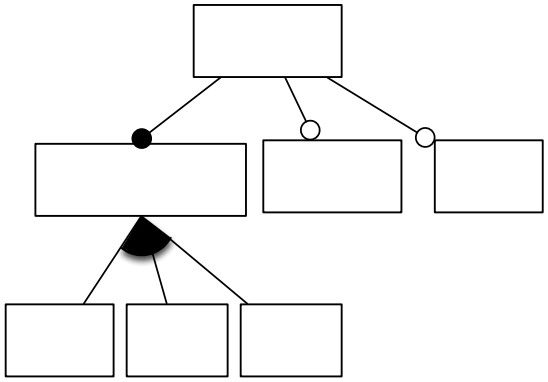
Let’s Hack a DSL!

http://tinyurl.com/HackDSLProject1718

50% of the final note

Hard, time-consuming but fun work!

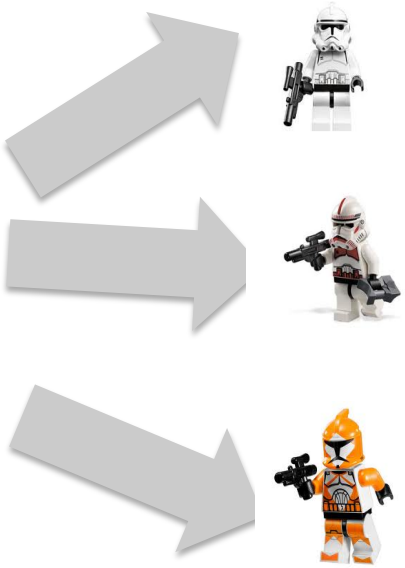
Let’s practice abstract/theoretical concepts



not, and, or, implies

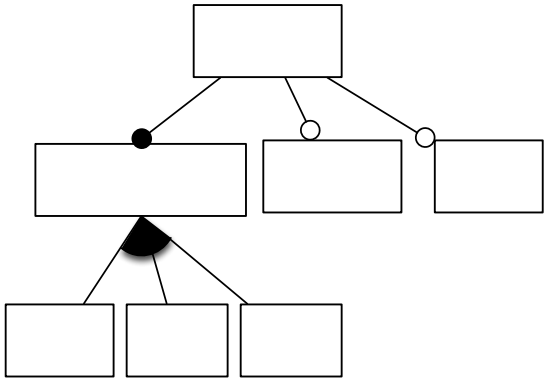
Feature models

(product lines)



Variants of code (e.g., Java or C)
Variants of user interfaces
Variants of video sequences
Variants of models (e.g., UML or SysML)
Variants of « things » (3D models)
…

x265 x264 FFmpeg



not, and, or, implies

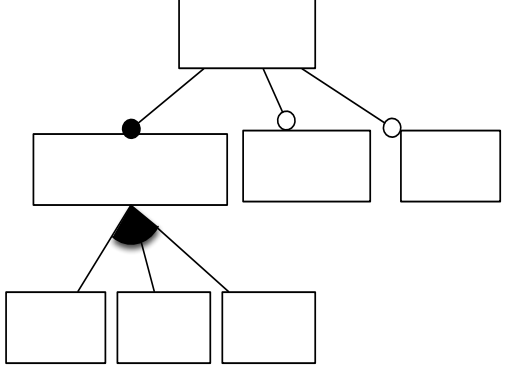
Feature models

solvers

configurators (UI)

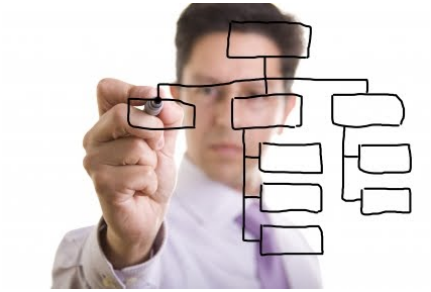
command line options

Feature Model

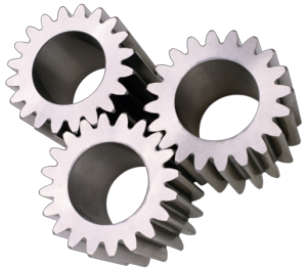


not, and, or, implies

Communicative



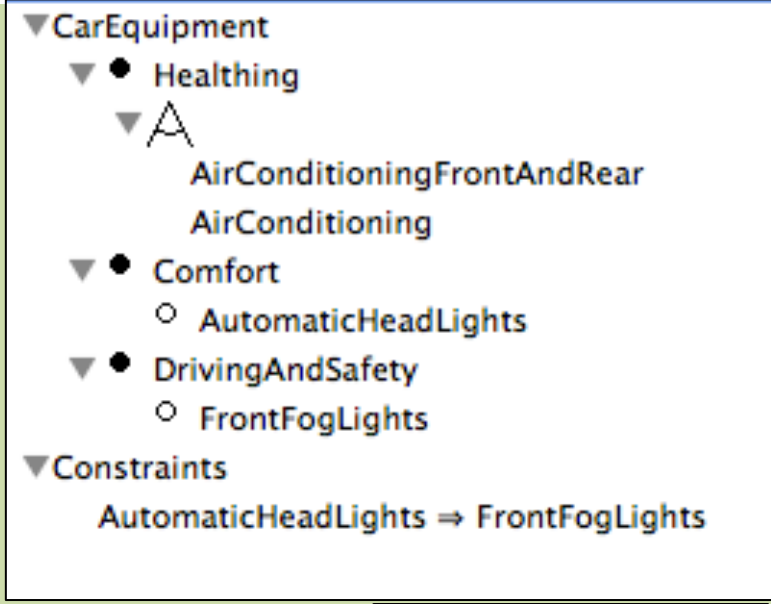
Analytic

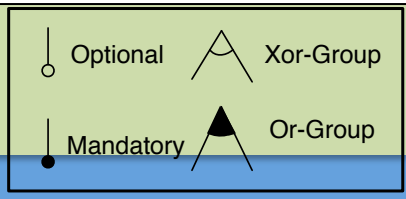


Generative

SPLC12Scripts Model | SPLC12Scripts.config

- CarEquipment (valid, 1 possible configurations)
  - Healthing
    - AirConditioningFrontAndRear
    - AirConditioning
  - Comfort
    - AutomaticHeadLights
  - DrivingAndSafety
    - FrontFogLights

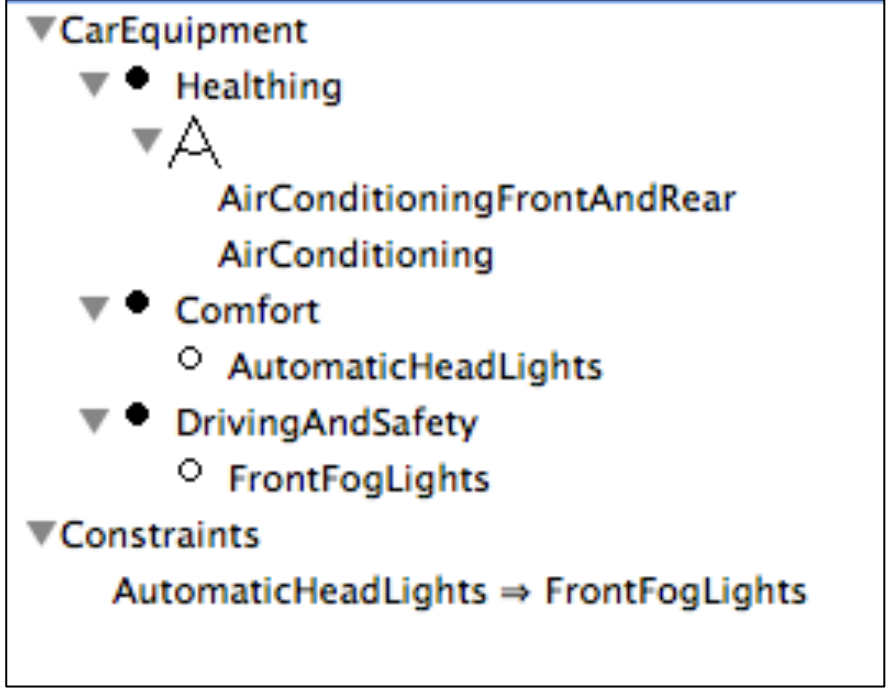


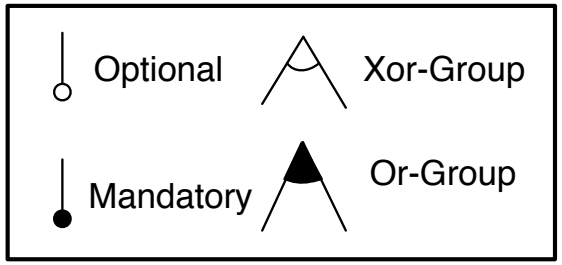


Hierarchy + Variability
=
set of valid configurations

configuration = set of features selected

{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning}





Definition 2 (Feature Diagram) *A feature diagram* $FD = \langle G, E_{MAND}, G_{XOR}, G_{OR}, I, EX \rangle$ *is defined as follows:* $G = (\mathcal{F}, E, r)$ *is a rooted, labeled tree where* $\mathcal{F}$ *is a finite set of features,* $E \subseteq \mathcal{F} \times \mathcal{F}$ *is a finite set of edges and* $r \in \mathcal{F}$ *is the root feature ;* $E_{MAND} \subseteq E$ *is a set of edges that define mandatory features with their parents ;* $G_{XOR} \subseteq \mathcal{P}(\mathcal{F}) \times \mathcal{F}$ *and* $G_{OR} \subseteq \mathcal{P}(\mathcal{F}) \times \mathcal{F}$ *define feature groups and are sets of pairs of child features together with their common parent feature ;* I *a set of implies constraints whose form is* $A \Rightarrow B$, EX *is a set of excludes constraints whose form is* $A \Rightarrow \neg B$ *(*$A \in \mathcal{F}$ *and* $B \in \mathcal{F}$*).*

Definition 3 (Feature Model) *An FM is a tuple* $\langle FD, \psi \rangle$ *where* FD *is a feature diagram and* ψ *is a propositional formula over the set of features* $\mathcal{F}$*.*

#1 Basic Feature Model and Xtext

- Elaborate an (Ecore) metamodel of feature modeling formalism
- Elaborate an Xtext grammar for specifying feature models (invent a concrete syntax!)
  - Write some examples (feature models) in your new language
- Compare the metamodel you have designed with the metamodel generated from your Xtext grammar

#2 Transformation of Feature Models

- Create a procedure for producing random feature models
- Transform feature models into:
  - DIMACS format for interoperating with SAT4J solver
  - JavaBDD solver
  - Minizinc for interoperating with some CSP solvers (eg Choco)
  - Z3 format for interoperating with Z3 solver
- Propose basic operations for feature model analysis
  - Checksing satisfiability of feature model, Core features, Dead features, False optional features, Enumeration of all configurations

based on solvers mentionned above!

- Benchmark/test solvers on random feature models

#3 From feature model to configurator

- Implement a basic user interface (UI) for presenting a configurator
- UI widgets: check boxes
  - three-states: selected, deselected, unselected
- Follow the hierarchy of the feature model for the UI
- Technology:
  - JavaFX or Swing
  - HTML/JS/CSS (harder)
- Plug one of the solver (see step #2) to propagate choices at each step of the configuration process

#4 From feature models to command line options

- Transform feature model into Command Line Options
  - Find two tools that manage command line options
  - Implement a transformation in these two specific technologies
- Questions:
  - How to transform constraints between features?
  - Comment on the gap between feature models and command line options (eg what cannot be expressed)
- Reading: Mikolás Janota, Fintan Fairmichael, Viliam Holub, Radu Grigore, Julien Charles, Dermot Cochran, Joseph R. Kiniry: CLOPS: A DSL for Command Line Options. DSL 2009: 187-210

#5 Attributed feature models

- Add the ability to add some “attributes” (integer domains) associated to some features
- #1, #2, #3, and #4 again
  - It is a new language => new Xtext grammar (#1)
  - reasoning => use CSP solvers (with Minizinc, #2)
  - configurator => new UI with “sliders”, #3
  - command line options => support for numerical options #4
- Discuss how you reuse your previous works with basic feature models

#6 Variability Model in the Real World

- Choose a real software project between ffmpeg, x264, and x265
- Elaborate a feature model of command line options with your DSL
- Generate a configurator from your feature model
- Generate command line options from your feature model
- Sample some configurations of your feature model and execute them