







Software Product Line Engineering

modeling and managing variability of software intensive systems

Dr. Mathieu Acher

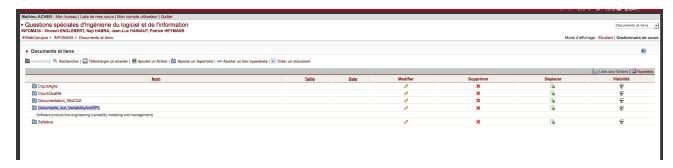
email: macher@fundp.ac.be

Prof. Patrick Heymans

University of Namur
PReCISE Research Centre

Material

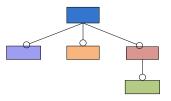
- http://www.fundp.ac.be/etudes/cours/ page_view/INFOM435/
 - Folder: "Documents_sur_VariabilityAndSPL"
 - Slides, exercises, evaluation



Email: macher@fundp.ac.be

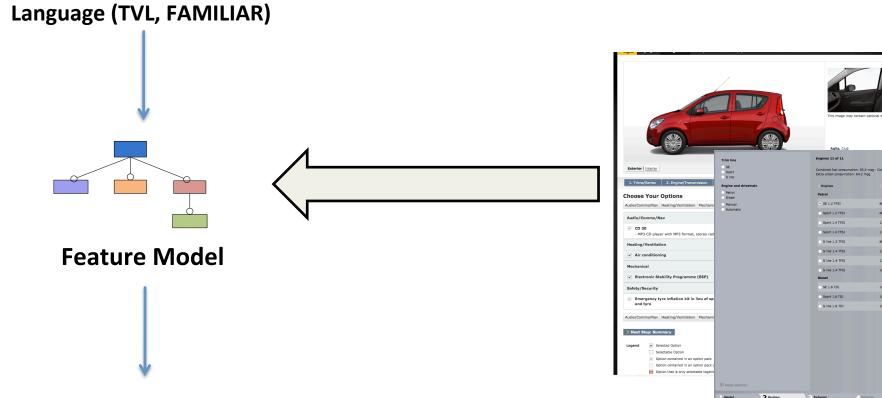
Previously

- Feature Models (FMs)
- Practice
 - Existing car configurators
 - TVL (FM language)
 - FAMILIAR (FM management)



Feature Model

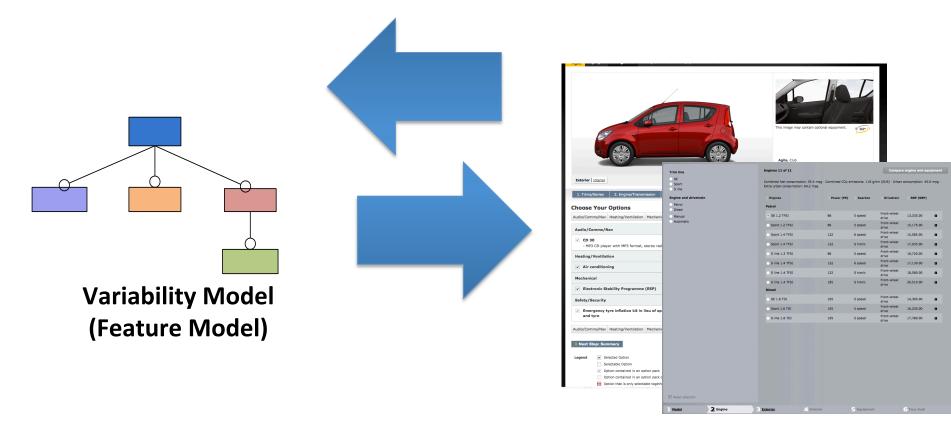
Previously





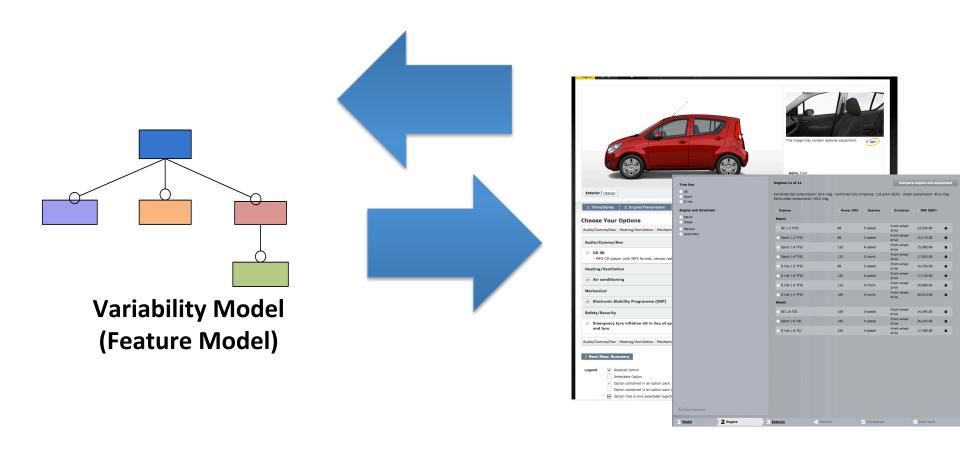
Automated reasoning

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



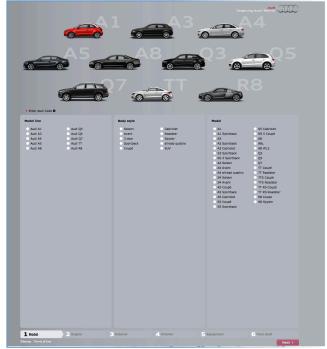
Today

- Back to our modeling experience
 - Feature model management (FAMILIAR)
 - Support to better understand and play with your specification
 - Existing techniques can be considered in your work (merging)
 - Building views (slicing)
- Project
 - Re-engineering configurators
 - « reverse engineering » + « engineering »
 - Now: From feature models to configurators
 - Model-based approach



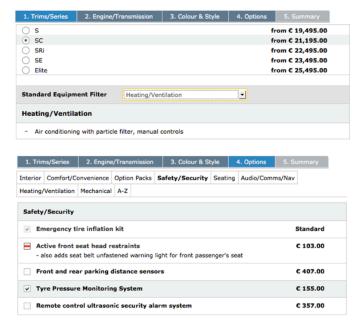
- 70% of the final mark
- Expected output
 - Configurator artefacts (source code, metamodels/models, demo)
 - Presentation
 - "defense"
- Good properties
 - Solution that works
 - Reusable artefacts
 - Customizable solution
 - Close to existing configurator ("re-engineering")
 - Or even better: "better"

- 1 configurator by group
 - the one you previously chose
- Expected
 - (mandatory) all configuration options in one place
 - (optional) "views"
 - Different containers / tabs / pages
 - Wizards (linear/sequential configuration process)



From feature models to configurators

- Configurator?
 - Graphical user interface (GUI)
 - Assist users
 - Select/deselect options
 - Propagation of choices

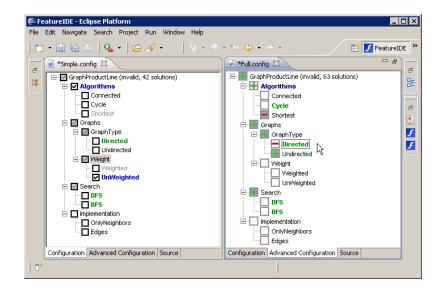


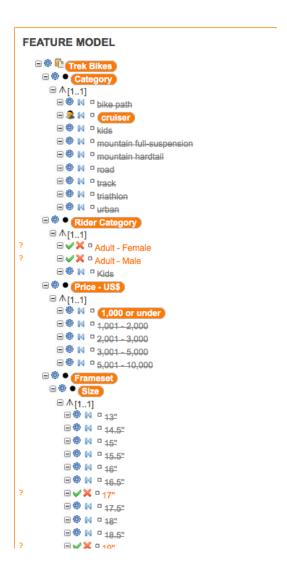
- Feature model?
 - Representation of the configuration "space"
 - Not a visually attractive solution!

Filling the gap

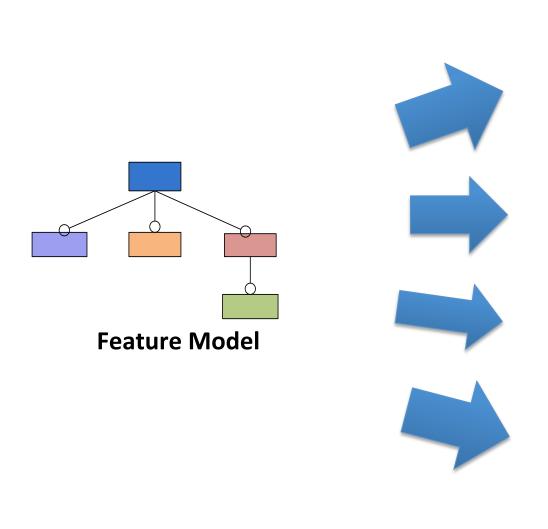
From feature models to configurators

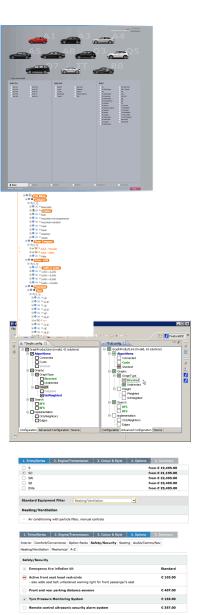
Something like this



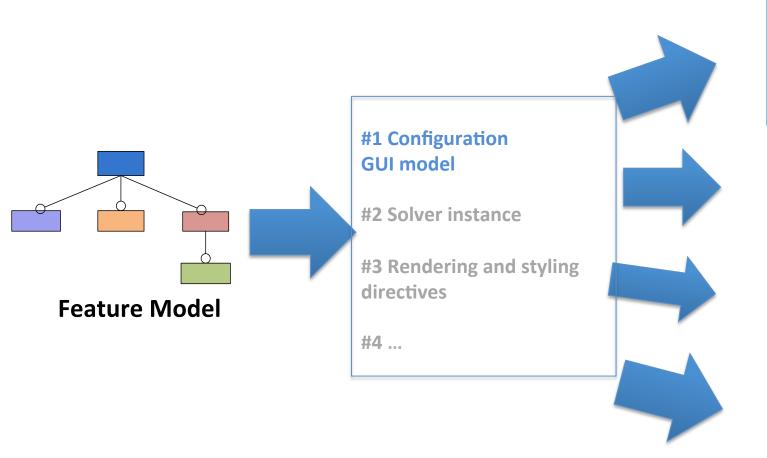


Generic, flexible solution

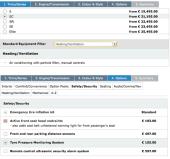




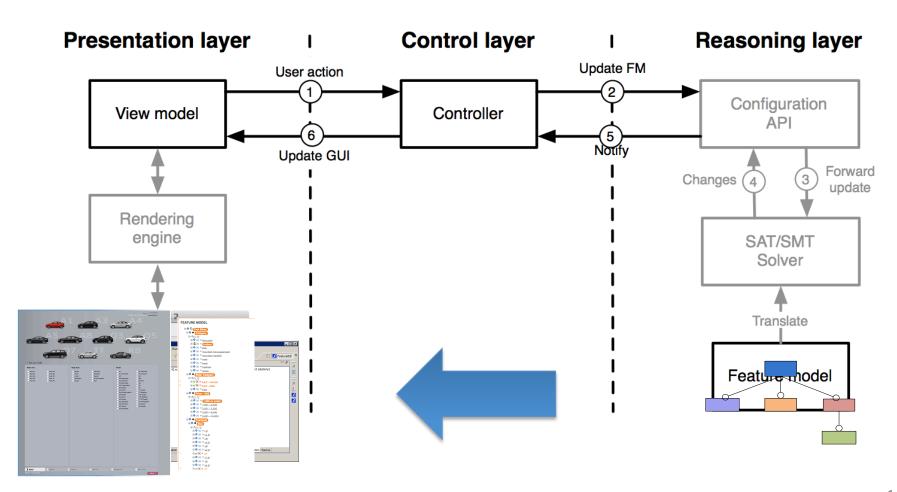
Model-based approach



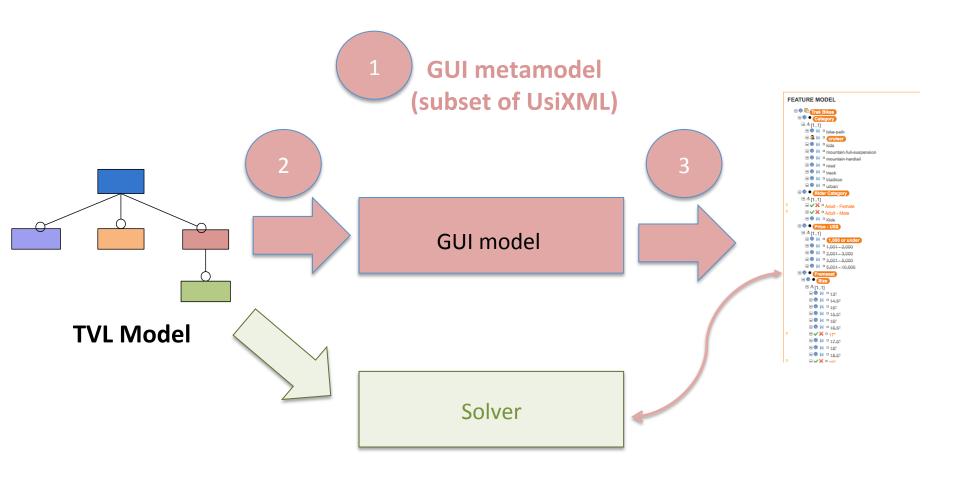




MVC Architecture



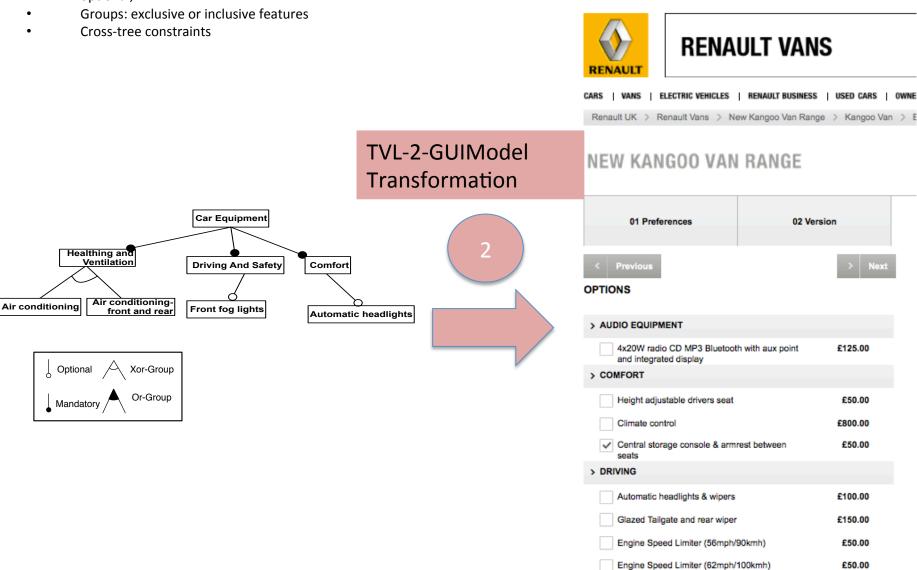
Model-based approach



Hierarchy: rooted tree

Variability:

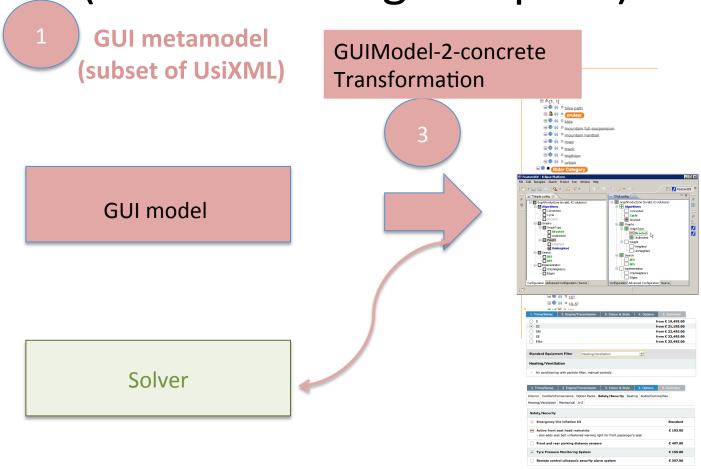
- mandatory,
- optional,



GUI metamodel

(subset of UsiXML)

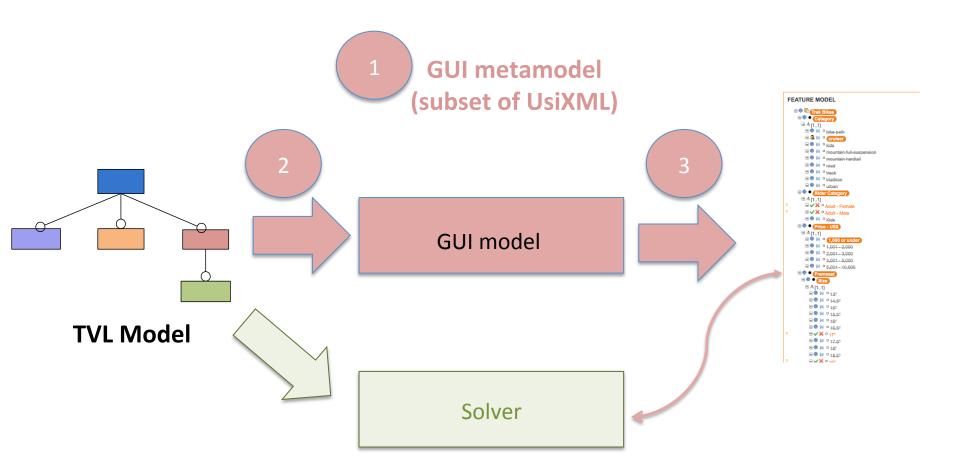
Concrete GUI (one technological space)



- Expected output
 - Configurator artefacts
 - TVL-2-GUIModel transformation
 - "generic rules"
 - API of TVL
 - metamodels/models
 - Analysis of your configurator and your technological space
 - Ideally: subset of UsiXML
 - Ideally(2): merge your efforts
 - GUIModel-2-concrete transformation
 - Connection with the solver
- Presentation
 - "Defense" (20' + 10' questions)
 - demo
 - Don't forget to report on "open issues" or "difficult problems"

- Good properties
 - Solution that works
 - Reusable artefacts
 - Customizable solution
 - Close to existing configurator ("re-engineering")
 - Or even better: "better"
- Be incremental
 - Simple metamodel, simple transformations
 - Working solution ASAP
 - then you can tackle more difficult problems
- Collaborative effort
 - Within the group
 - 3 main steps... but inter-related
 - (optional) Between the groups
 - Identify similarities (metamodel?) / reusable artefacts

Today



Ecore metamodel (Eclipse modeling)

Suggestion: all members should work on a first version of the metamodel (MM) Simple MM first, then one member of the group focusing on UsiXML