## DSM-TP 2016

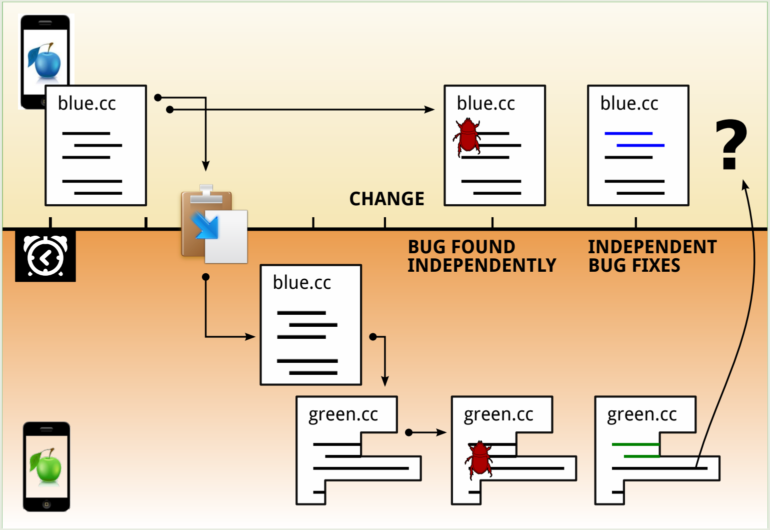
# Modeling Variability

Andrzej Wąsowski

VARIETE, DFF SAPERE AUDE

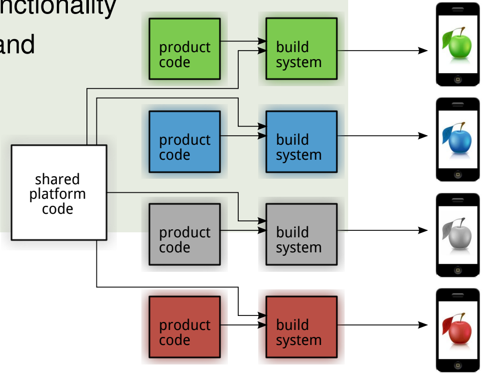
VARIES cost MPM4CPS

# Drowning in Clone-And-Own



# Opportunistic Reuse Does Not Work

- Common scenario:
  - version the code, reuse when **opportunity** appears
- If the file to be reused needs change, **copy** it
  - You *clone-and-own* it
- **Benefit** from **quickly** available functionality
- But have to test, debug, change and evolve the file **yourself**
- **Product specific** code grows
- Platform code **diminishes and degrades**



# SUCCESSFUL REUSE IS

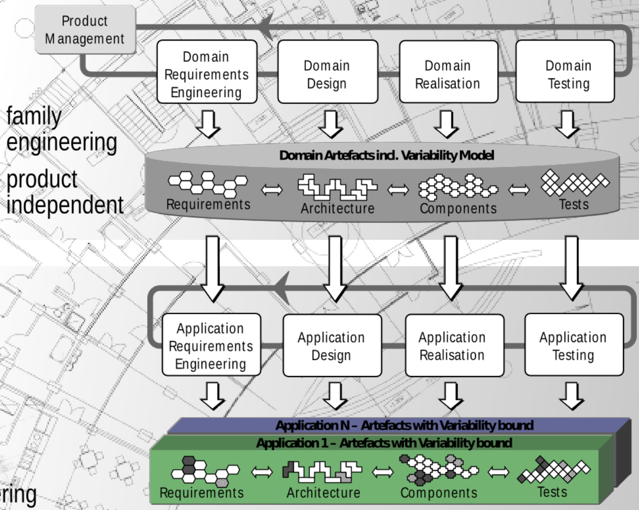
# PROACTIVE
# PLANNED
# MANAGED

- SPL Method, **Architecture**
- Variability **Implementation Spectrum**
- Variability Abstraction: **Feature Modeling**
- Variability Modeling in **Practice**
- Variability **Realization**

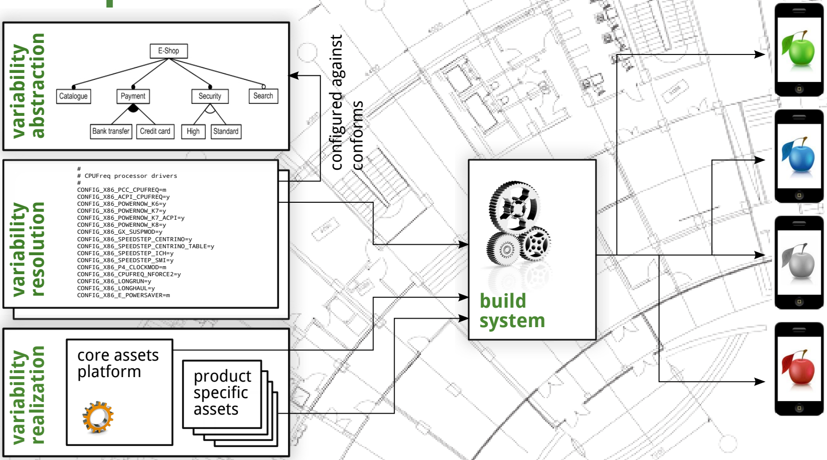
# AGENDA

# Domain vs Application Processes

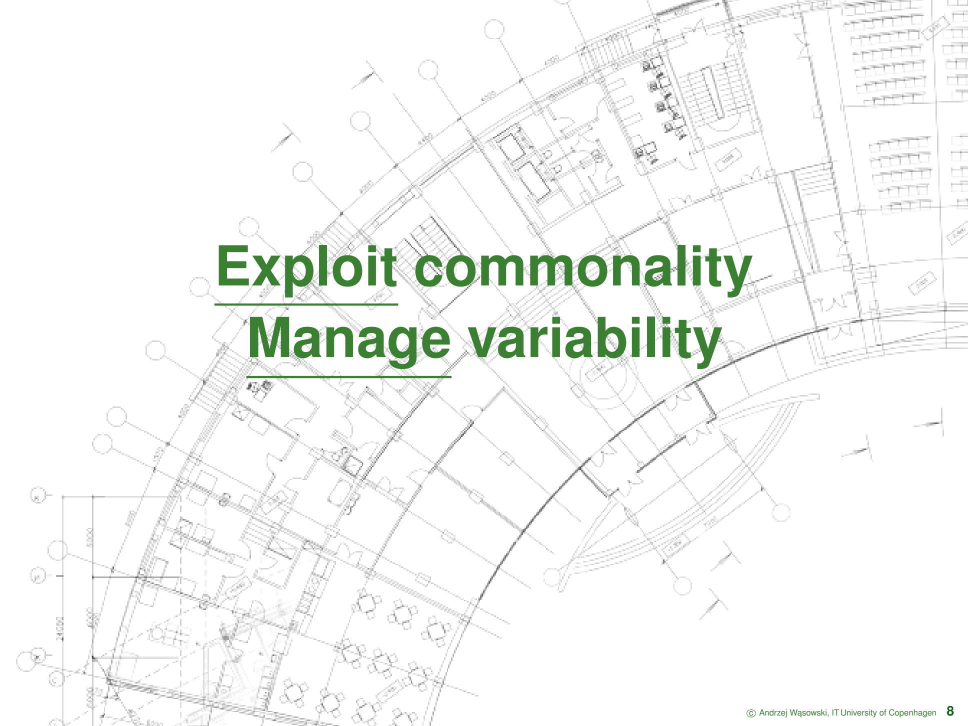
Pohl et. al. Software Product Line Engineering



# A Simple Product Line Architecture




- Less product specific = more reuse: development/tests/debugging/build
- Model of *commonality* and *variability*.
- Scope under control. Explicit feature life cycle



# Exploit commonality
# Manage variability

- SPL Method, **Architecture**
- Variability **Implementation Spectrum**
- Variability Abstraction: **Feature Modeling**
- Variability Modeling in **Practice**
- Variability **Realization**

# AGENDA

# Spectrum of Variability Architectures

## Stay as close to the left as possible

only product specific code (no reuse)

frameworks + framework completion code

domain specific languages + code generation

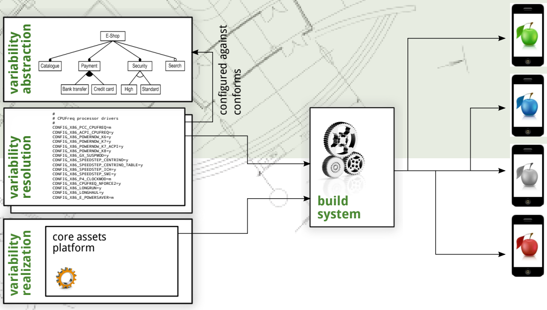
feature models + product specific code

feature models + build system

property & configuration files + build system

# Implementation Technologies

- Variability abstraction: FMs, DSLs, or **none**
- Variability resolution:
  - XML property file
  - FM configuration
  - Domain specific model (DSM)
- Variability realization:
  - general purpose code
  - w/ variability techniques
  - code generators
  - model transformers
  - parts may use DSLs
  - etc.



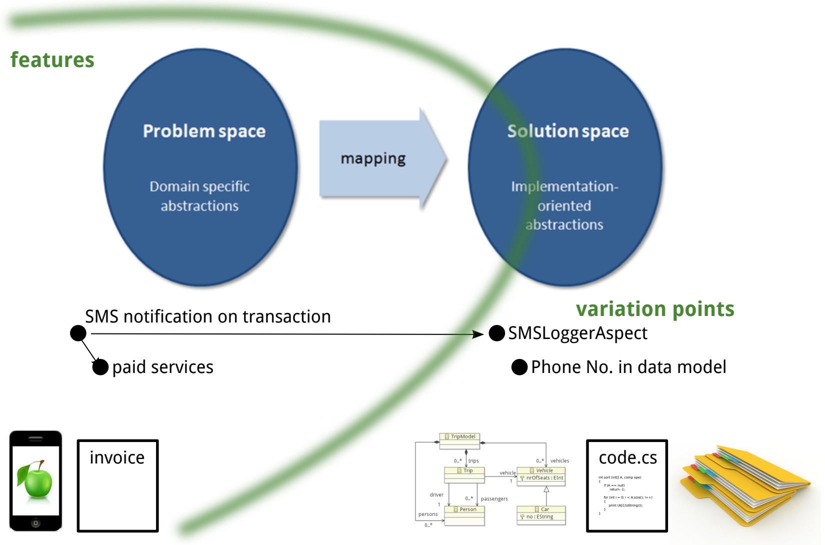
- SPL Method, **Architecture**
- Variability **Implementation Spectrum**
- Variability Abstraction: **Feature Modeling**
- Variability Modeling in **Practice**
- Variability **Realization**

# AGENDA

# Problem Space

# Solution Space



# CVL Architecture for Dummies

The degree of coupling can be controlled by moving the mapping

**Variability Abstraction** — Feature/Decision Models

**Variability Realization** — Feature Mapping

**Base (model)** — Source Code

CVL submitters. *Common Variability Language, OMG Revised Submission*. 2012

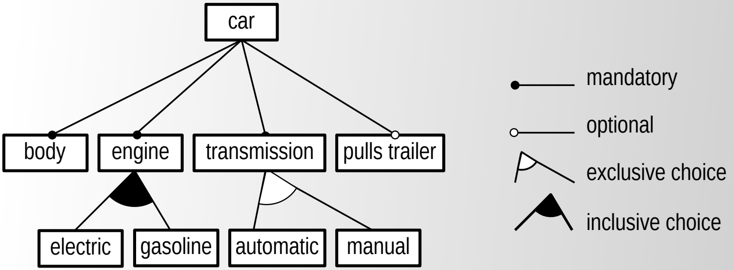
# Feature Modeling (I)

**feature:**
a single variability increment in the problem domain (decision)

**variation point:**
a single variability increment in the solution space

# Feature Modeling (II)

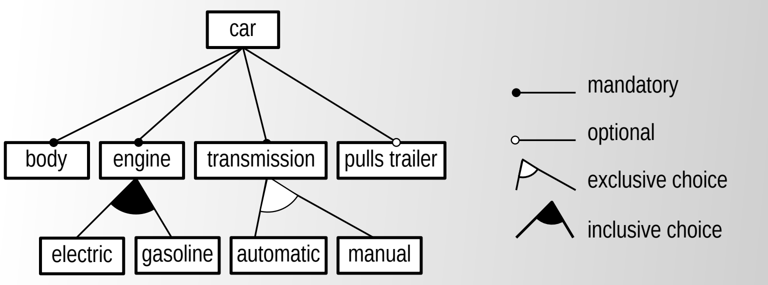
## Example from Czarnecki'02



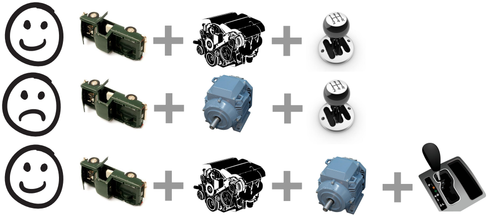

- **Hierarchy** constraints, for example:
  - manual requires transmission (each child node requires its parent node)
- **Groups** constraints: engine is electric or gas driven or both
- Not all constraints in hierarchy & groups, cross-tree constraints in text:
  - electric *requires* automatic
- Attributes are added like to classes (eg. engine volume)

# Feature Modeling (III)

## Configuration



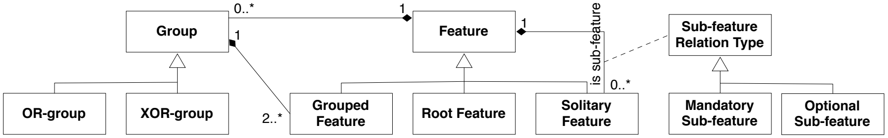

electric *requires* automatic



# Feature Models (IV)

## An example meta-model from Janota'08

meta-model (abstract syntax)




- Note a single generic kind of relations: **subfeature**
- No distinction between **kind-of** (inheritance) and **part-of** (containment), like class modeling does
- A characteristic feature of **configuration and constraint languages** (as opposed to structural modeling languages)
- Clafer (as a structural modeling langauge) supports the distinction, but so do other feature modeling languages

# Feature Modeling and FODA

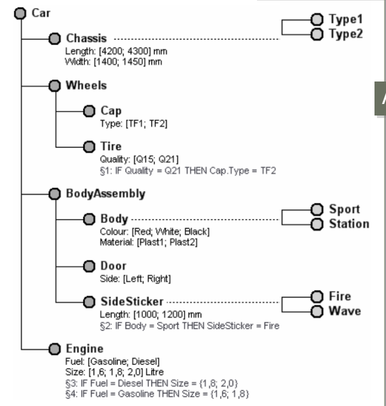
Feature Oriented Design and Analysis by Kang et al. 1990

- **FODA succeeds for its simplicity**
- Probably best intro in Czarnecki's *Generative Programming* (Chpt. 4)
- **3950+** citations, **never formally published**

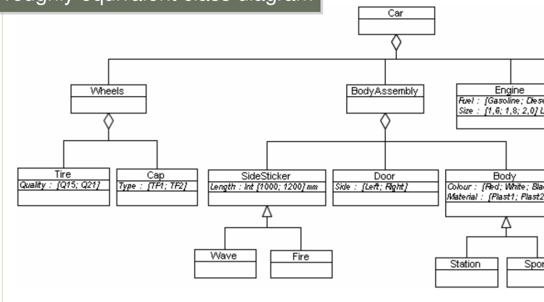
# Feature Modeling vs Class Modeling

## A feature model in *Product Variant Master* Notation (Hvam)

A feature model




A roughly equivalent class diagram




**More on this:** Bąk. Czarnecki. Wąsowski. *Feature and Meta-Models in Clafer: Mixed, Specialized, and Coupled*. SLE 2010

**Above models from:** Haug. Degn. Poulsen. Hvam. *Creating a documentation system to support the development and maintenance of product configuration systems*

# Applications of Feature Models

Design & Management

domain
modeling

product line scoping
product line mngmt

code generation
driving build system

driving
testing

Development & Test

# How To Build Feature Models?

Two strategies, but only one good :)

**top-down**

- **Big-bang** adoption
- Perform **careful domain analysis**
- Document **concepts, abstractions and relations** between them in a FM

**bottom-up**

- Identify a **cloned component**
- Find the **patches** that describe differences
- Translate **diffs to variation points**
- Organize **variation points into features**, and a hierarchy
- Works well with **incremental adoption**
- See SPLC07 paper by Danfoss

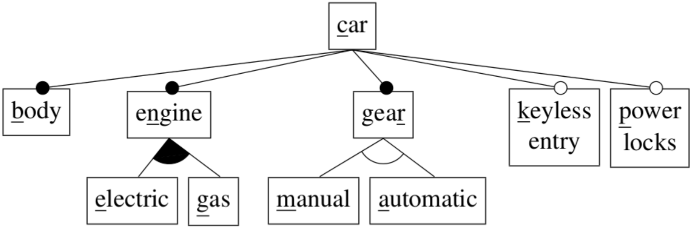
Hans Peter Jepsen, Jan Gaardsted Dall, Danilo Beuche. *Minimally Invasive Migration to Software Product Lines.* SPLC 2007

- SPL Method, **Architecture**
- Variability **Implementation Spectrum**
- Variability Abstraction: **Feature Modeling**
- Variability Modeling in **Practice**
- Variability **Realization**

# AGENDA

# Variability Modeling is The Success Story of Modeling

# A Laboratory Feature Model

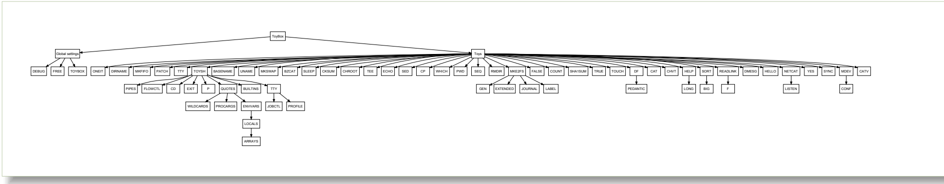



$$keyless\_entry \rightarrow power\_locks$$

Czarnecki, Wąsowski. *Feature Diagrams and Logics: There and Back Again.* In: 11th International Software Product Line Conference (SPLC 2007) Kyoto, Japan, 10-14 September, 2007 © IEEE Press.

# A Healthy Wild Feature Model Cub

## ToyBox project, 71 features




**The Linux Kernel has 6-12K features, depending how you count!**
**But maximum depth is 8, most leaves are at 4!**

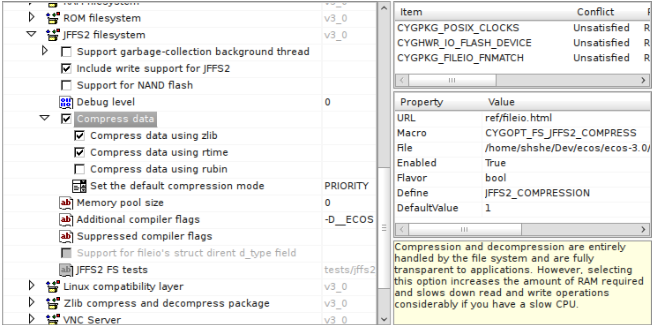
↓ this is the Linux kernel model fit to the slide width ↓

---


Berger, She, Lotufo, Wąsowski, Czarnecki. *A Study of Variability Models and Languages in the Systems Software Domain.* IEEE Transactions in Software Engineering, 2013

# Is FODA special? Not really!

eCos configurator




- **Linux kernel** and **eCos operating system** use similar configurator, controlled by **textual variability models**
- **Trees become unwieldy** very fast
- Many tools used **linearized trees**, like above
- Nice **trees are good for PowerPoint**, whiteboard and brainstorming

# It is easy to design a textual syntax

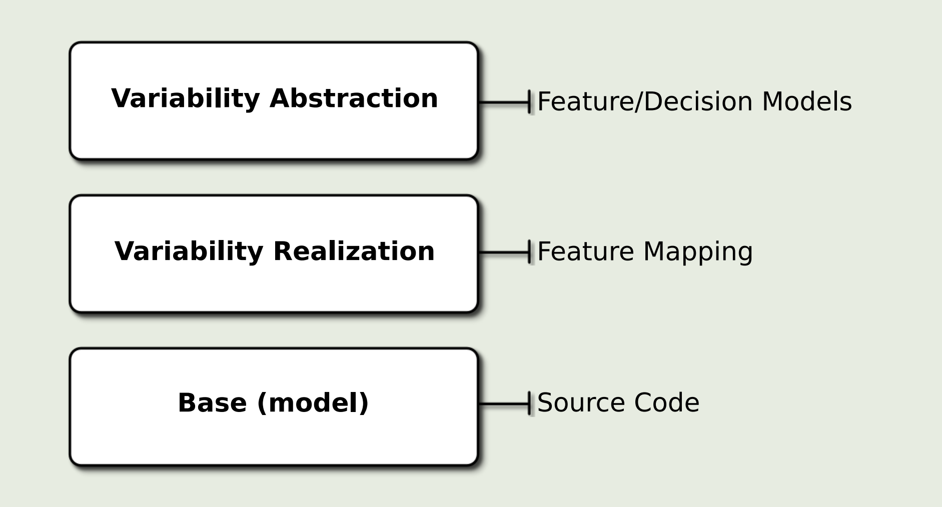
Kconfig of Linux kernel, designed by non-experts, with no tools
CDL of eCos, designed by non-experts, with no tools

```
menuconfig MISC_FILESYSTEMS
  bool "Miscellaneous filesystems"

 if MISC_FILESYSTEMS

  config JFFS2_FS
   tristate "Journalling Flash File System" if MTD
   select CRC32 if MTD




 config JFFS2_FS_DEBUG
  int "JFFS2 Debug level (0=quiet, 2=noisy)"
  depends on JFFS2_FS
  default 0
  range 0 2
  --- help ---
    Debug verbosity of ...


 config JFFS2_FS_WRITEBUFFER
  bool
  depends on JFFS2_FS
  default HAS_IOMEM

```

```
cdl_component MISC_FILESYSTEMS {
 display    "Miscellaneous filesystems"
 flavor     none
}

cdl_package CYGPKG_FS_JFFS2 {
  display    "Journalling Flash File System"
  requires   CYGPKG_CRC
  implements CYGINT_IO_FILEIO
  parent     MISC_FILESYSTEMS
  active_if  MTD

  cdl_option CYGOPT_FS_JFFS2_DEBUG {
   display          "Debug level"
   flavor           data
   default_value    0
   legal_values     0 to 2
   define           CONFIG_JFFS2_FS_DEBUG
   description      "Debug verbosity of...."
  }

  cdl_option CYGOPT_FS_JFFS2_NAND {
   flavor           bool
   define           CONFIG_JFFS2_FS_WRITEBUFFER
   calculated       HAS_IOMEM
  }
```

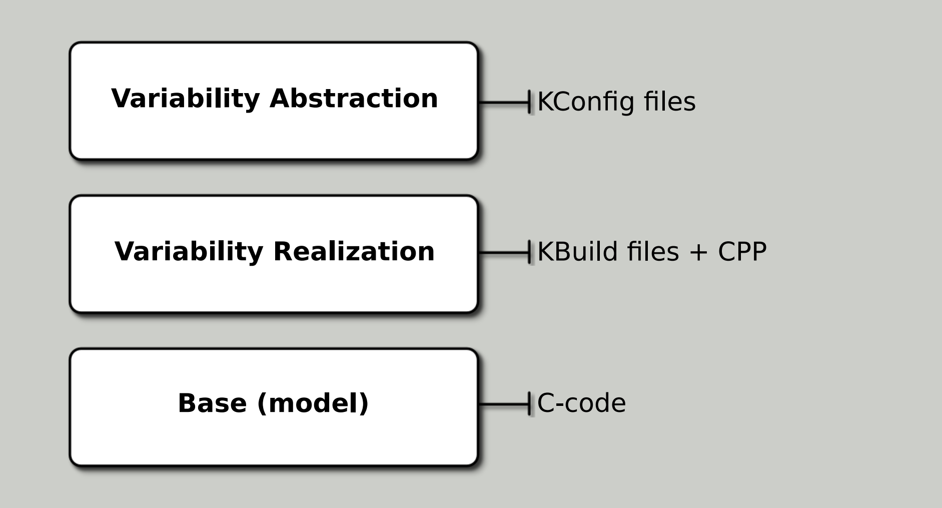
# CVL Architecture for Dummies

The degree of coupling can be controlled by moving the mapping

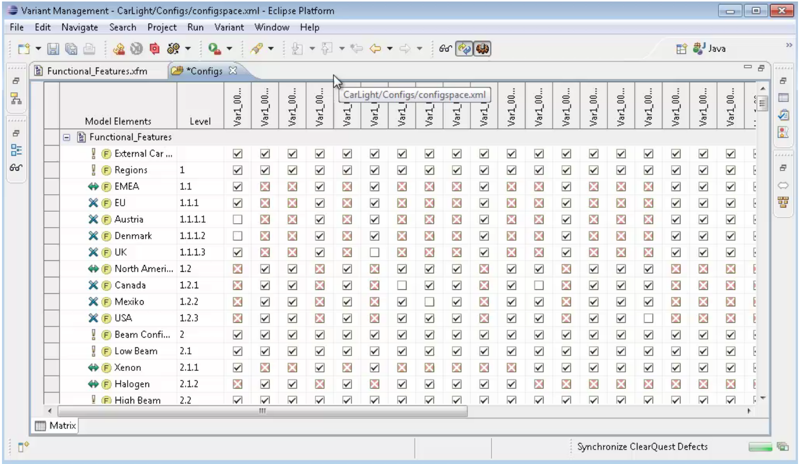



CVL submitters. *Common Variability Language, OMG Revised Submission*. 2012

# CVL Architecture for Linux Junkies



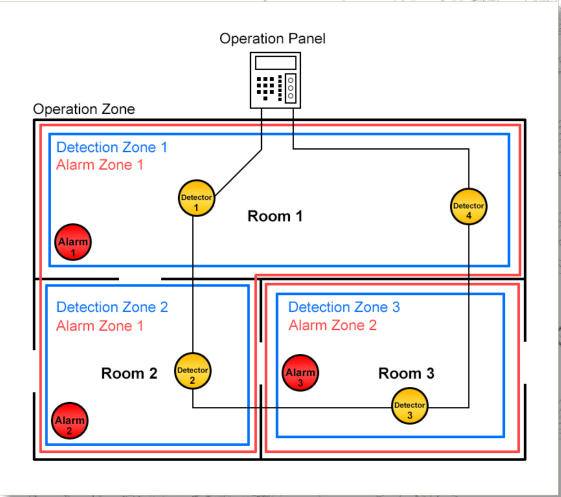
# Grid View in Pure•Variants




- Commercial tools support **multiple views** of the same model
- Some vendors: Pure Systems (DE), Big Lever (US), most PLM tools
- Clafer also has the grid view

# Are all variability models trees?

## A Fire Alarm System

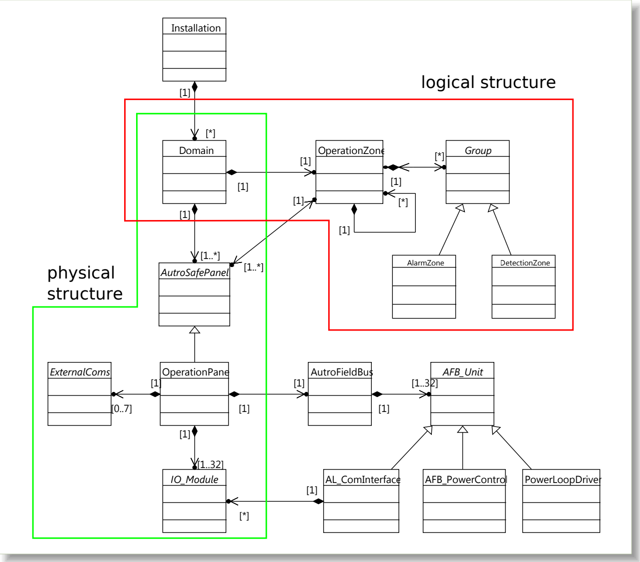



- detection zones
- alarm zones
- wiring
- three different structures
- Modeled as constrained class diagrams!
- Sometimes called **topological variability**

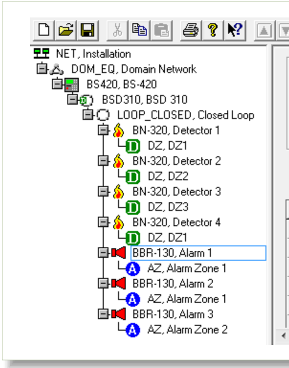
Berger. Stanciulescu. Øgård. Haugen. Larsen. Wąsowski. *To connect or not to connect: experiences from modeling topological variability.* SPLC 2014
Fantechi. *Topologically configurable systems as product families.* SPLC 2013

# Modeled using class diagrams in Papyrus

## A model view showing two model hierarchies




A home grown **configurator** used to instantiate models

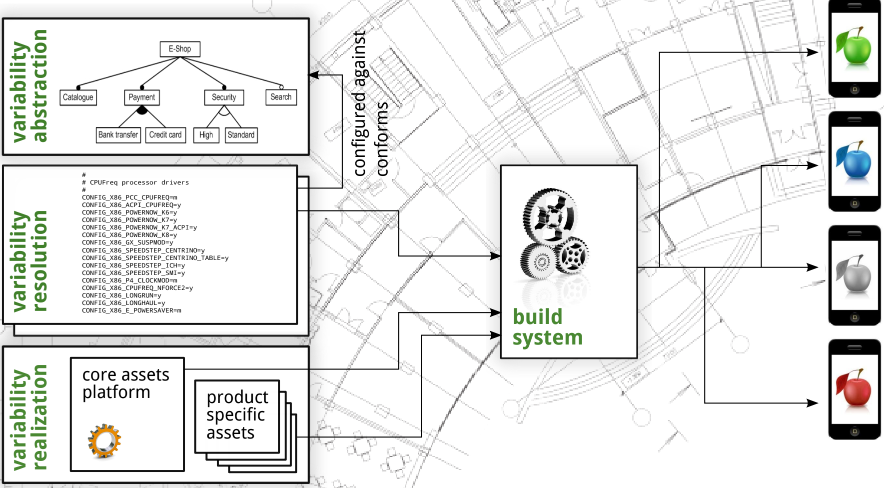


- SPL Method, **Architecture**
- Variability **Implementation Spectrum**
- Variability Abstraction: **Feature Modeling**
- Variability Modeling in **Practice**
- Variability **Realization**

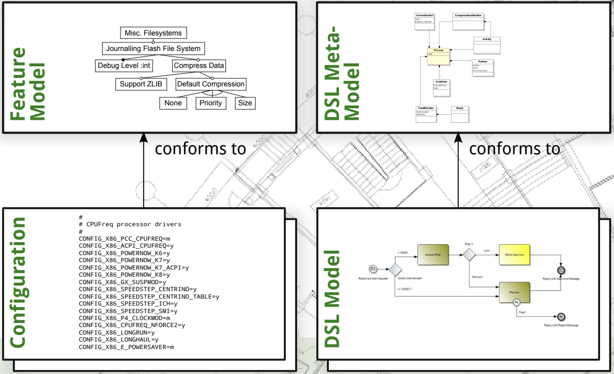
# AGENDA

# Connect Abstraction to Realization

## Most of the school is about it :)



# Feature Models vs DSLs

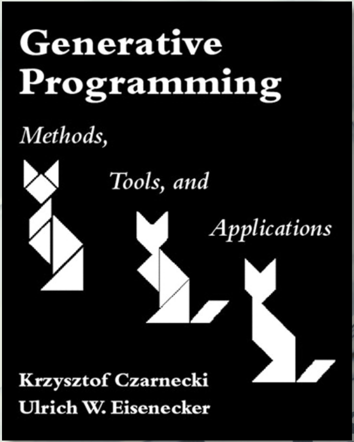


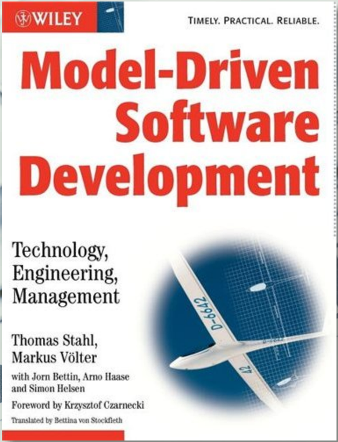

- Feature models are **ready and simple** (no design effort, deep insight)
- DSL requires **design effort**, but rewards with more **expressiveness**
- Effort also translates to **maintenance**
- FM effort is offset by **existing feature modeling tools**
- DSL development effort is offset by **language workbenches**

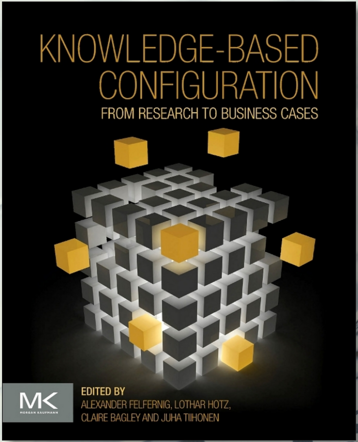
# Advice on Realization

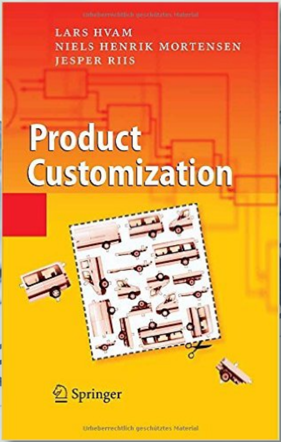
[Stahl and Völter]

- Choose **functional** domain **concepts** as features / DSL concepts
- Start a **small** domain model and **grow** it **iteratively**
- Keep the **build automatic** at all times
- Generate/synthesize **legible code/models**
- We follow these principles in the Clafer tutorial on railway stations


Generative Programming
Methods,
Tools, and
Applications
Krzysztof Czarnecki
Ulrich W. Eisenecker


WILEY
TIMELY. PRACTICAL. RELIABLE.
Model-Driven Software Development
Technology, Engineering, Management
Thomas Stahl, Markus Völter
with Jorn Bettin, Arno Haase and Simon Helsen
Foreword by Krzysztof Czarnecki
Translated by Bettina von Stockfleth
D-6642


KNOWLEDGE-BASED CONFIGURATION
FROM RESEARCH TO BUSINESS CASES
EDITED BY
ALEXANDER FELFERNIG, LOTHAR HOTZ, CLAIRE BAGLEY AND JUHA TIIHONEN


LARS HVAM
NIELS HENRIK MORTENSEN
JESPER RIIS
Product Customization
Springer

- SPL Method, **Architecture**
- Variability **Implementation Spectrum**
- Variability Abstraction: **Feature Modeling**
- Variability Modeling in **Practice**
- Variability **Realization**

# AGENDA