Software Product Line Testing Part II : Variability Modeling

Myra Cohen Matthew Dwyer

Laboratory for Empirically-based Software Quality Research
Department of Computer Science
University of Nebraska - Lincoln

Work supported by NSF CCF through awards 0429149 and 0444167, by the U.S. Army Research Office through award DAAD190110564 and by an NSF EPSCoR FIRST award.

e^2 ESQuaReD

University of Nebraska Lincoln

Outline

Software Product Lines : What and Why?

- Modeling Variability in Software Product Lines

Validating Product Lines

A Framework for Variability Coverage

Toward Product Line Driven Test Processes

Outline

- Modeling Variability in Software Product Lines

1. What is variability?
2. Variability and other attributes
3. Feature models
4. Rich variability modeling notations
5. A formal variability modeling framework

What is Variability?

Commonality

The features shared by a set of systems

Variability

The features that differ between some pair of systems

Variability as an Abstraction

Mechanisms for implementing variability

- Compile flags
- Properties files
- Command-line arguments
- Inheritance
- Interface definition (and information hiding)
- Design patterns (e.g., strategy)
- Connectors (e.g., in architecture)

We are interested in the abstraction

Honda Sedan Variability

- Model : Civic, Accord
- Package : Sedan, Coupe, Hybrid, GX, Si
- Transmission : manual, auto, cvt
- Power : gas, hybrid, natural gas
- Doors : 2, 4
- Cylinders : 4, 6
- Nav system : Y/N
- ABS

Types of Variability

External Variability

– Visible to the customer:
  - Example: manual vs automatic transmission
  - Example: your cell phone may or may not have a camera and you may have different resolution options

Internal Variability

– Hidden from customer:
  - Example: battery technology in hybrid electric car
  - Example: communication protocol

Product Line = Variability

Variability is the key concept in product lines

A product line with no variability
is a single system

To define a product line we must define
the ways that instances of the product line
may vary

Defining Variability

Lots of terminology in the literature

feature, variation, variability, …

We will use Pohl et al.'s terminology

variation point

- A feature of PL instances that may vary

variant

- The realization of a feature

dependence

- Declares the potential binding of a realization to a feature

Honda Sedan

Variation Points

- model, package, transmission, power, doors, cylinders

Variants

- Civic, Accord, gas, hybrid, natural, gas, 2, 4

Dependences

- Model either Accord or Civic
- Nav system is optional
- ABS is mandatory

Variability & Development Artifacts

Variability must be expressed in …

requirements

architecture

design

implementation

testing …

in a coordinated manner.

Avionics Mission Computing

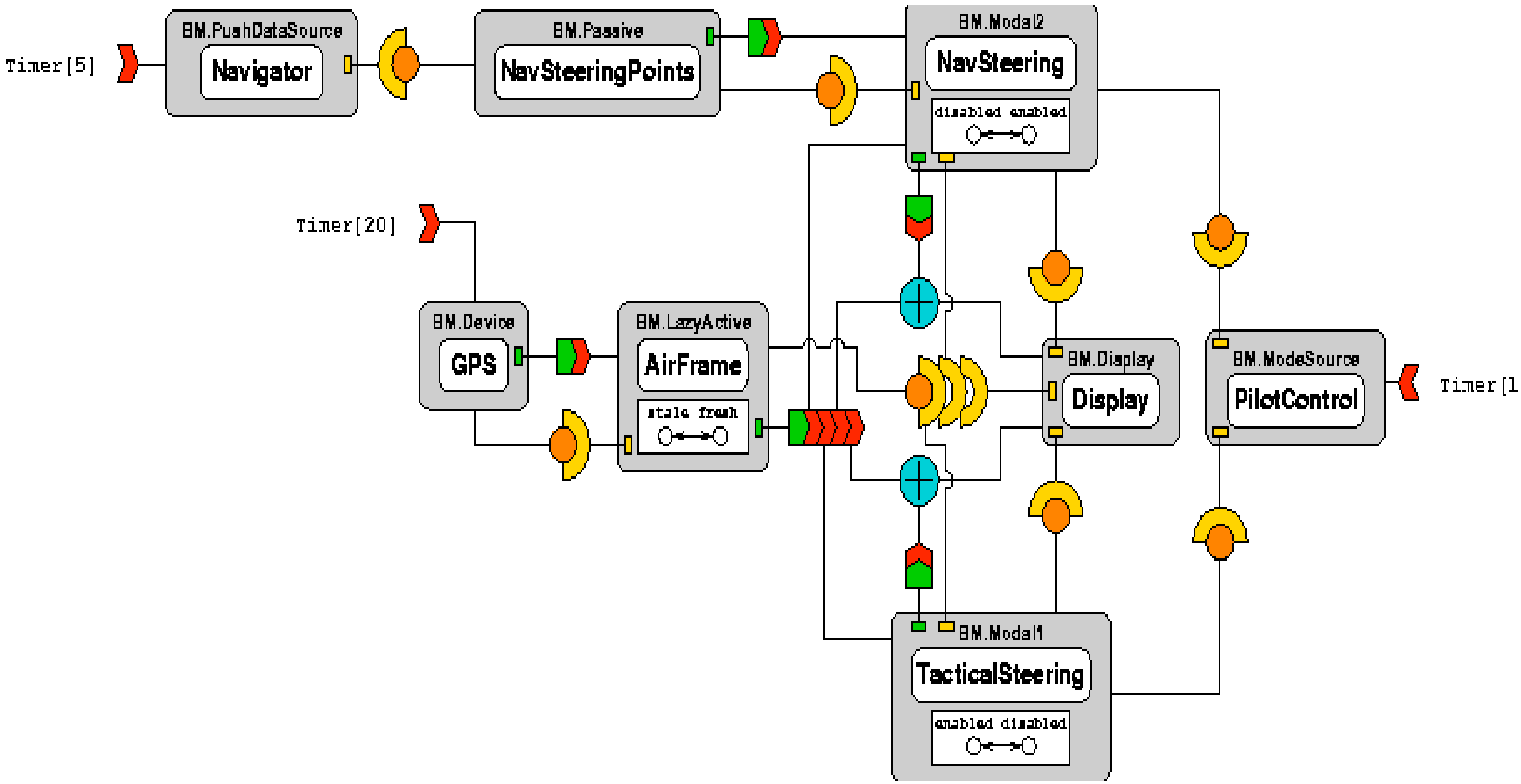
Enormous range of aircraft and missions

Enormous space of requirements and feature variability

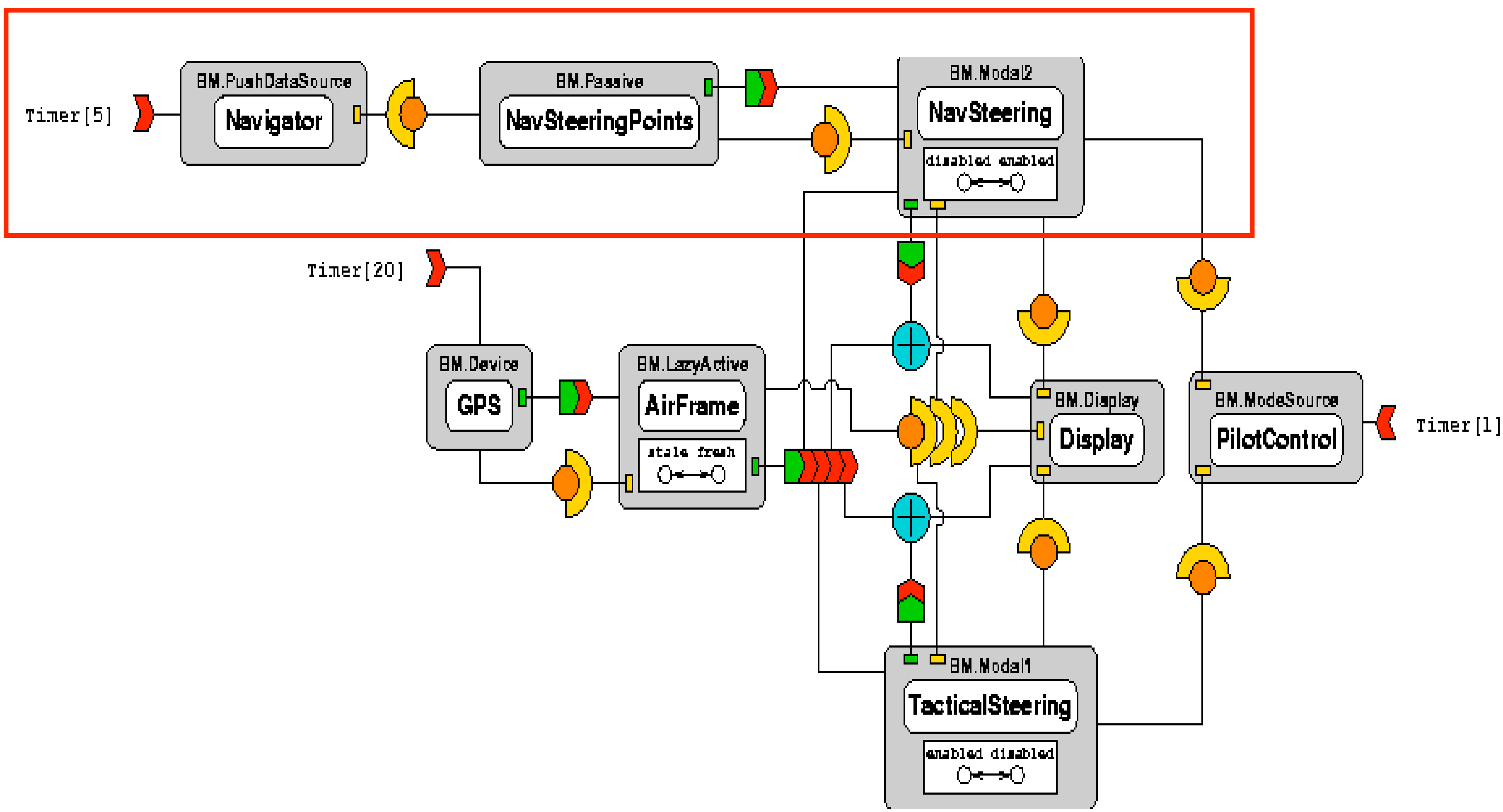
Consider autopilot navigation

- Requirements : it is required or not
- Architecture : include components and integrating connectors for auto-navigation facilities with rest of system

CADENA Component Architecture for Modal Steering



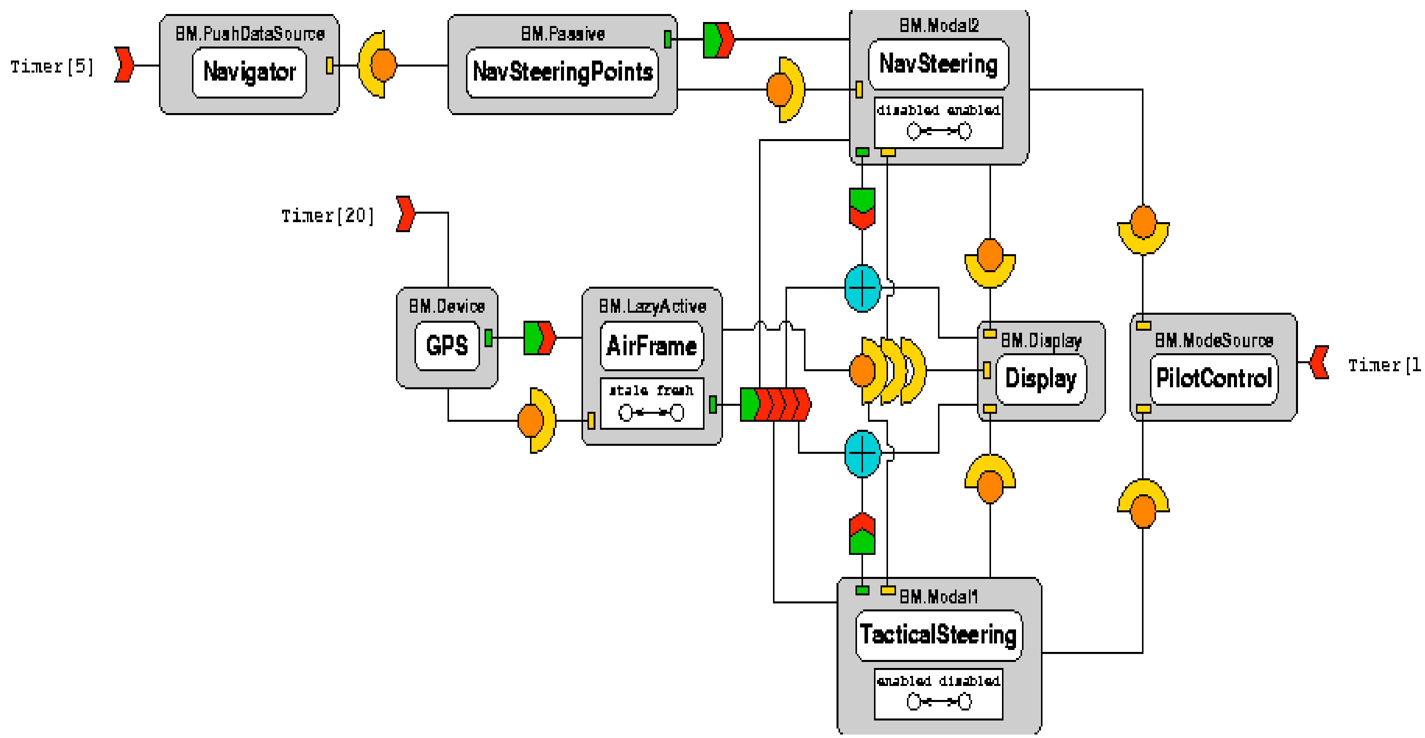
Optional Autopilot Navigation Subsystem (Feature)



Coordinating Variability

Requirements: Auto-navigation is present in system

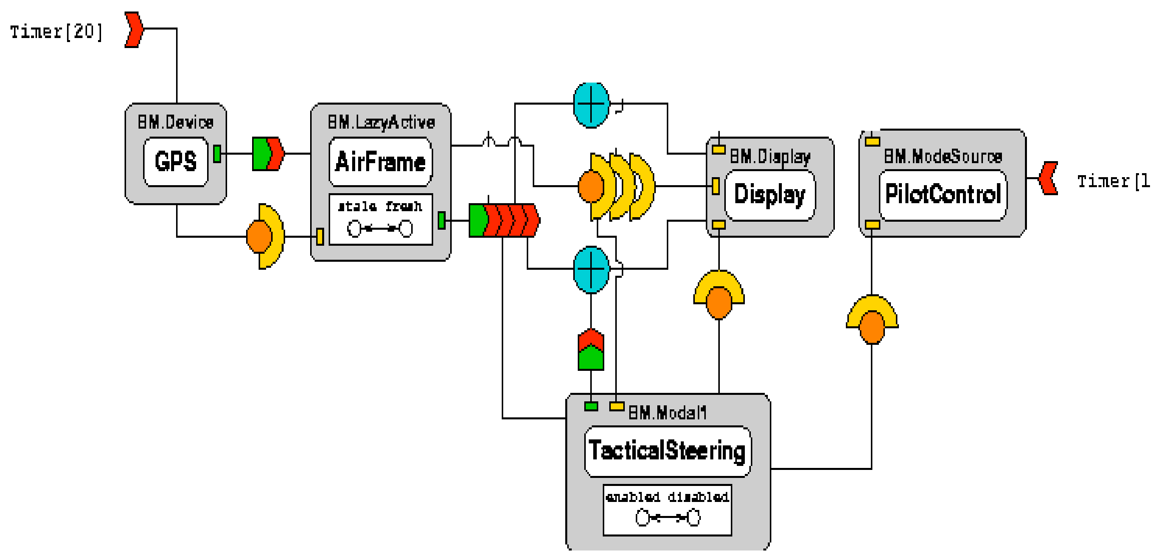
Architecture:



Coordinating Variability

Requirements: Auto-navigation is **not** present in system

Architecture:



Modeling Product Lines

An important aspect of successful product line development is defining an architecture that enables systematic reuse

We need a way to model the architectural details in order to represent the variability and commonality

Feature Oriented Domain Analysis

SEI FODA Project in Late 1980s

Identified features (variability) as the key to software product lines

Identified the need for artifact-independent modeling of the features in an SPL

Introduced the feature diagram

Feature Diagrams

Trees of features

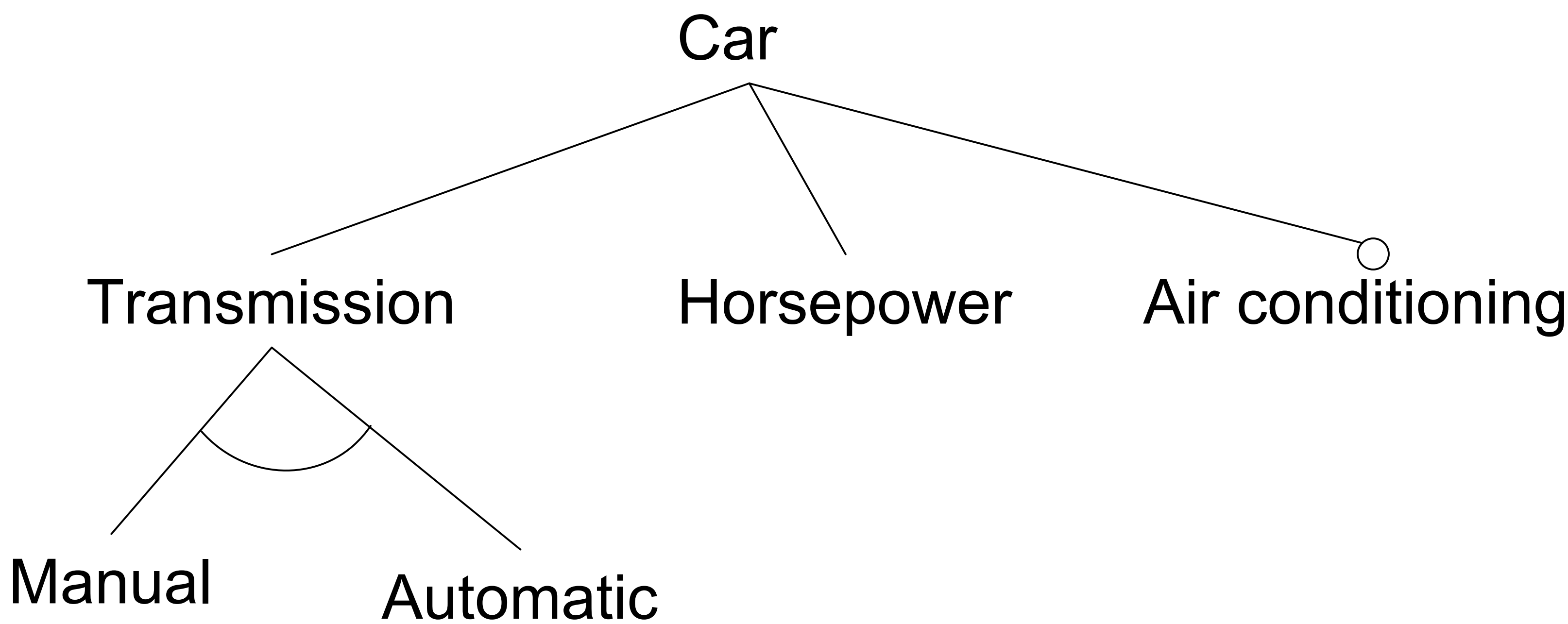
- Nodes represent variation points and variants

Child relationship represents binding

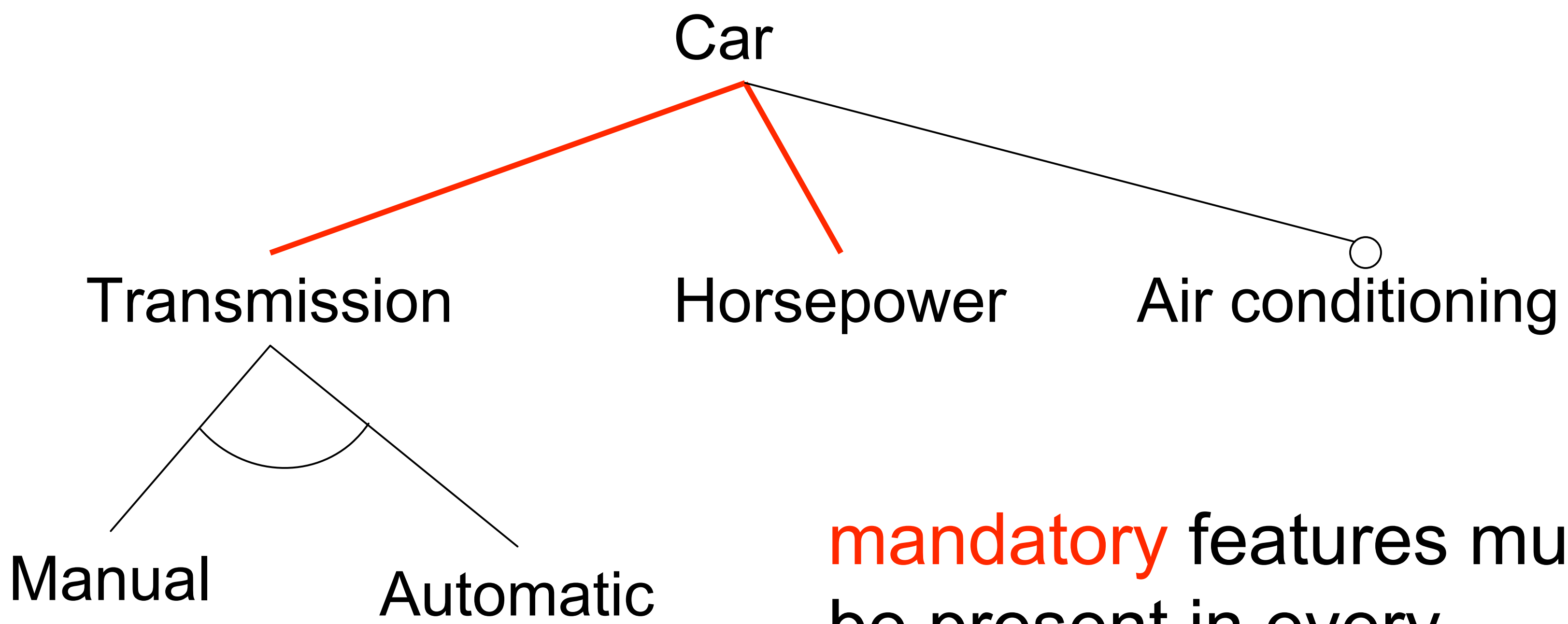
- Dependence

and/or graphs provide flexibility in defining feature realizations/relationships

FODA Feature Diagram Example

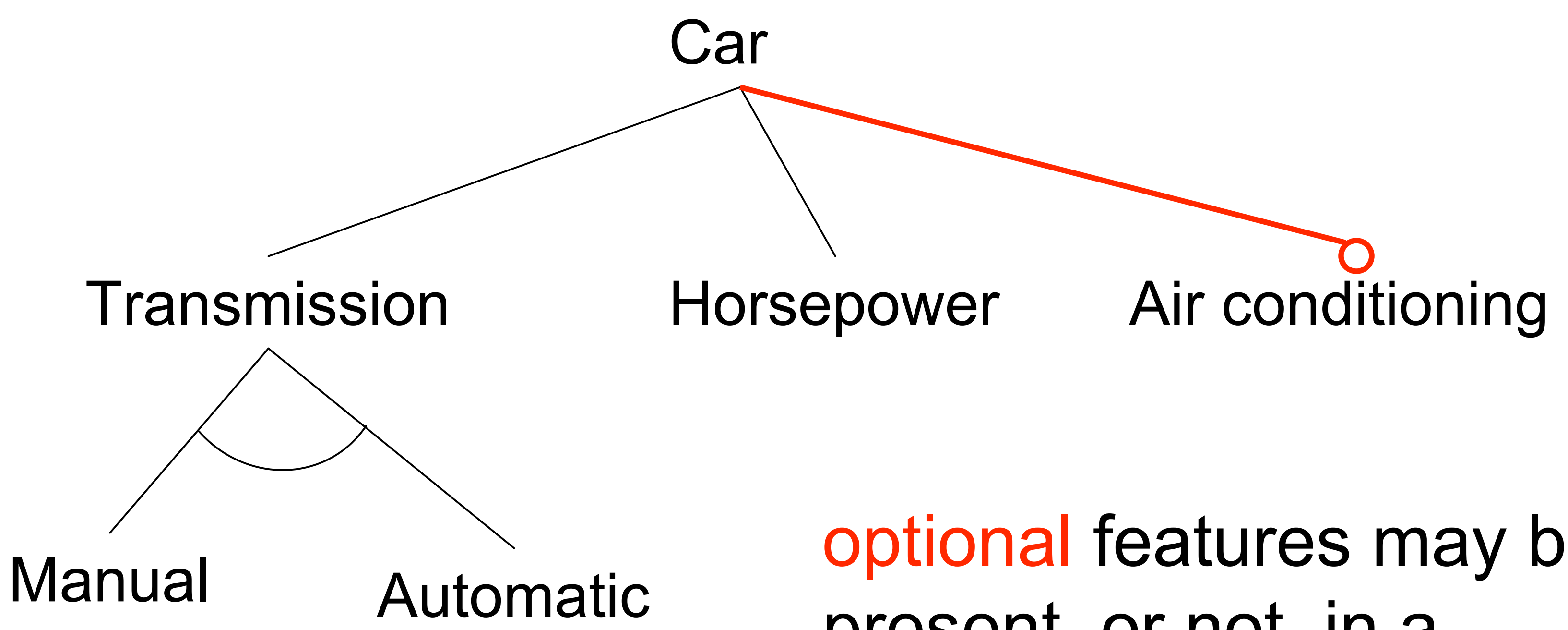


FODA Feature Diagram Example



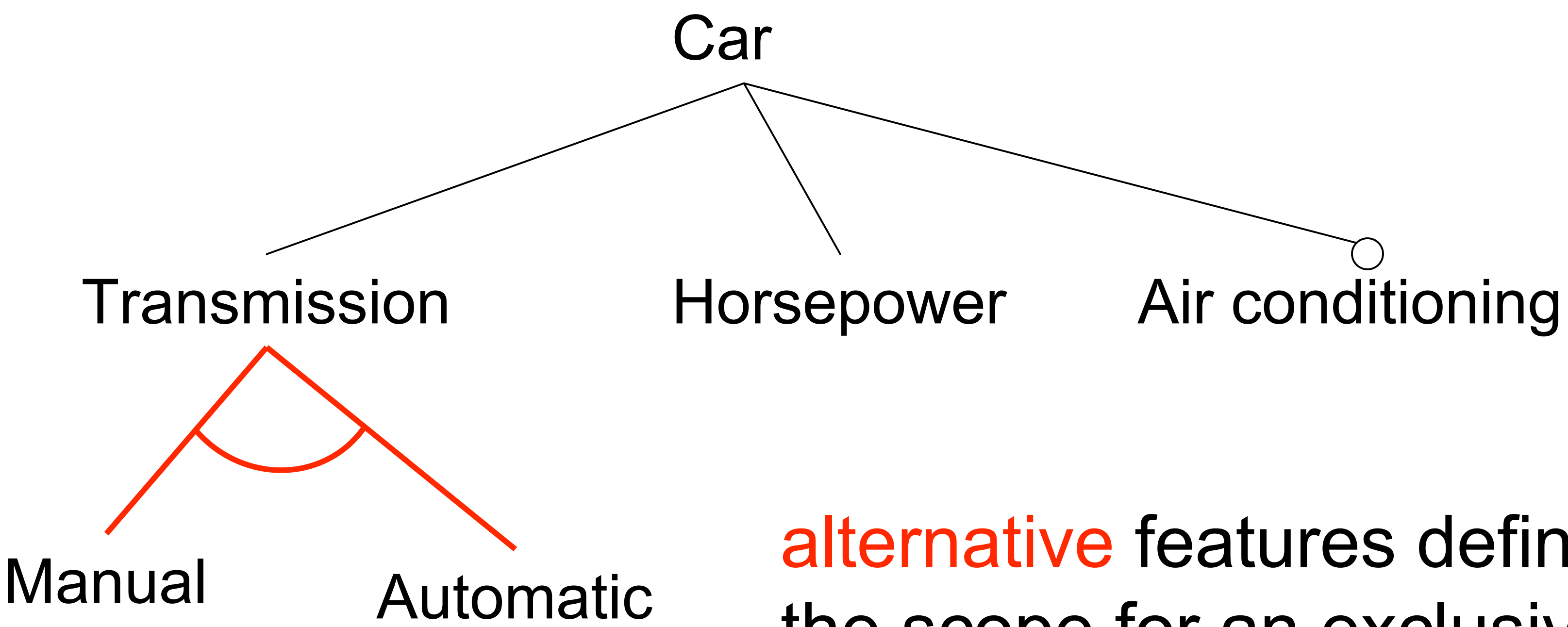
mandatory features must be present in every product line instance

FODA Feature Diagram Example



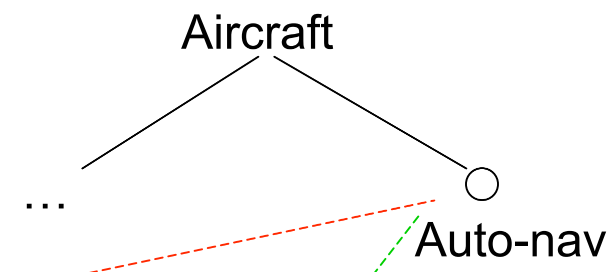
optional features may be present, or not, in a product line instance

FODA Feature Diagram Example



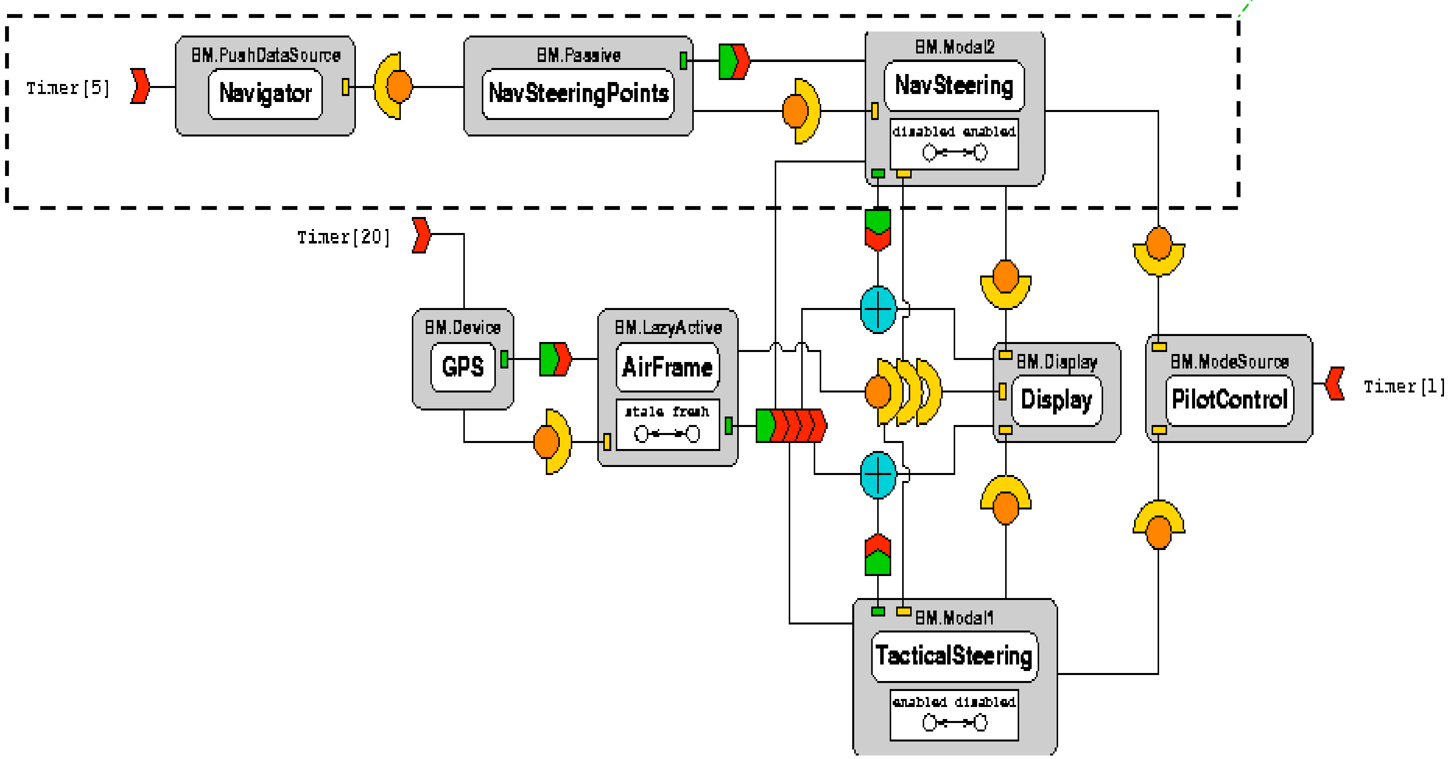
alternative features define the scope for an exclusive-or choice of features

Coordinating Variability



Requirements: Auto-navigation is not present in system

Architecture:



Constraints

Not all possible combinations of features correspond to feasible SPL instances

FODA introduced simple composition rules

- feature1 requires feature2
- feature3 excludes feature4

Constraints are essential for defining complex SPLs

feature diagram + constraints = feature model

Honda Sedan Constraints

Model:Civic excludes Cylinders:6

(Package:Coupe or Package:Si) requires Doors:2

Package:GX requires Power:natural gas

Package:Hybrid requires Power:hybrid

Package:Hybrid excludes Transmission:auto

(Model:Accord and Cylinders:6) requires Nav system

Building on FODA

In recent years, several efforts have extended feature model constraint languages

We focus on two such extensions

- Czarnecki et al.'s cardinality-based models
- Pohl et al.'s orthogonal variability model (OVM)

Cardinality-based Feature Models

Feature models cannot express the multiplicity of features present in a PL instance

For example

- A car has between 3 and 12 cylinders
- An airplane can have between 1 and 6 engines

Multiplicity of features is an essential point of variation in product lines

Cardinality-based Feature Models

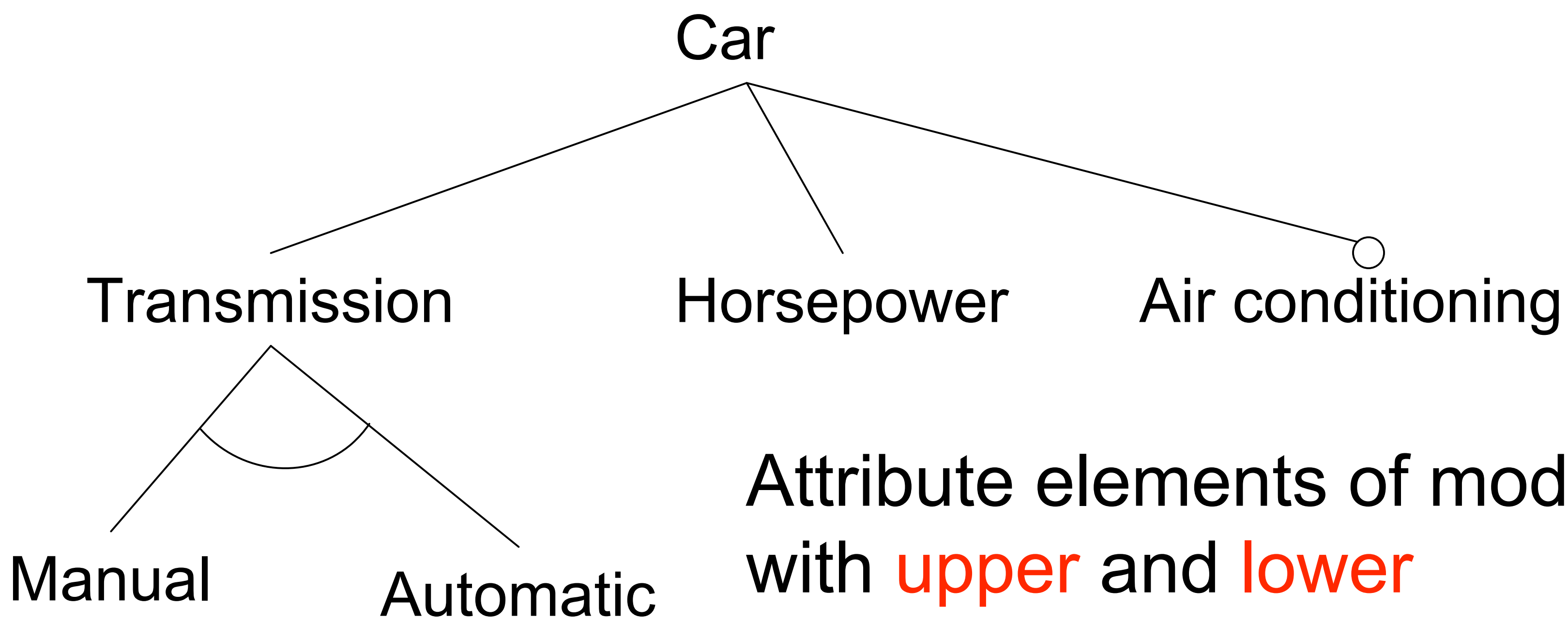
Cardinality constraints can be associated with all of the attributes of a feature model

Variation Points
- e.g., the number of seats in a car

Variants
- e.g., multiple sensors to guard against hardware failure

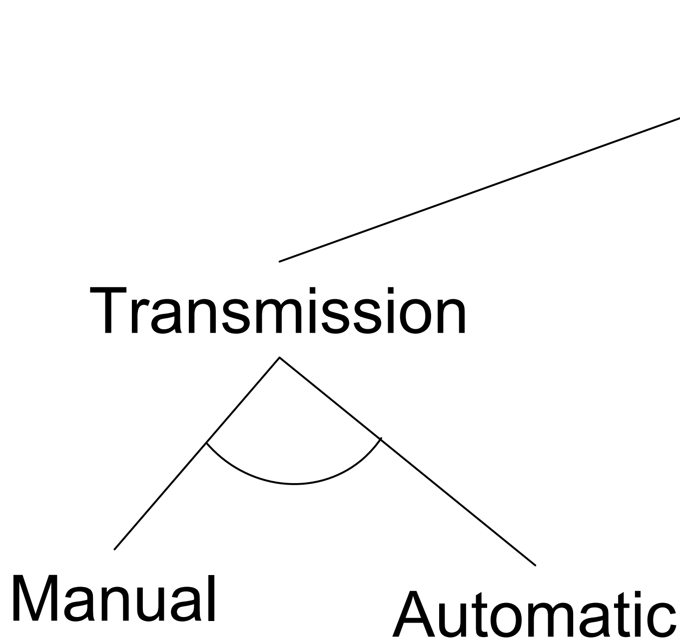
Dependences
- e.g., multiple music players radio, cd, mp3

Implicit FODA Cardinalities



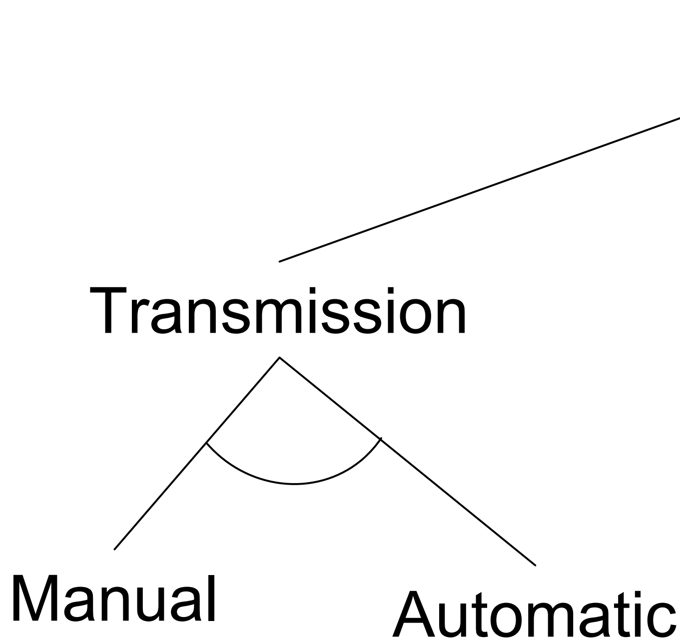
Attribute elements of model with upper and lower bounds on multiplicity

Implicit FODA Cardinalities



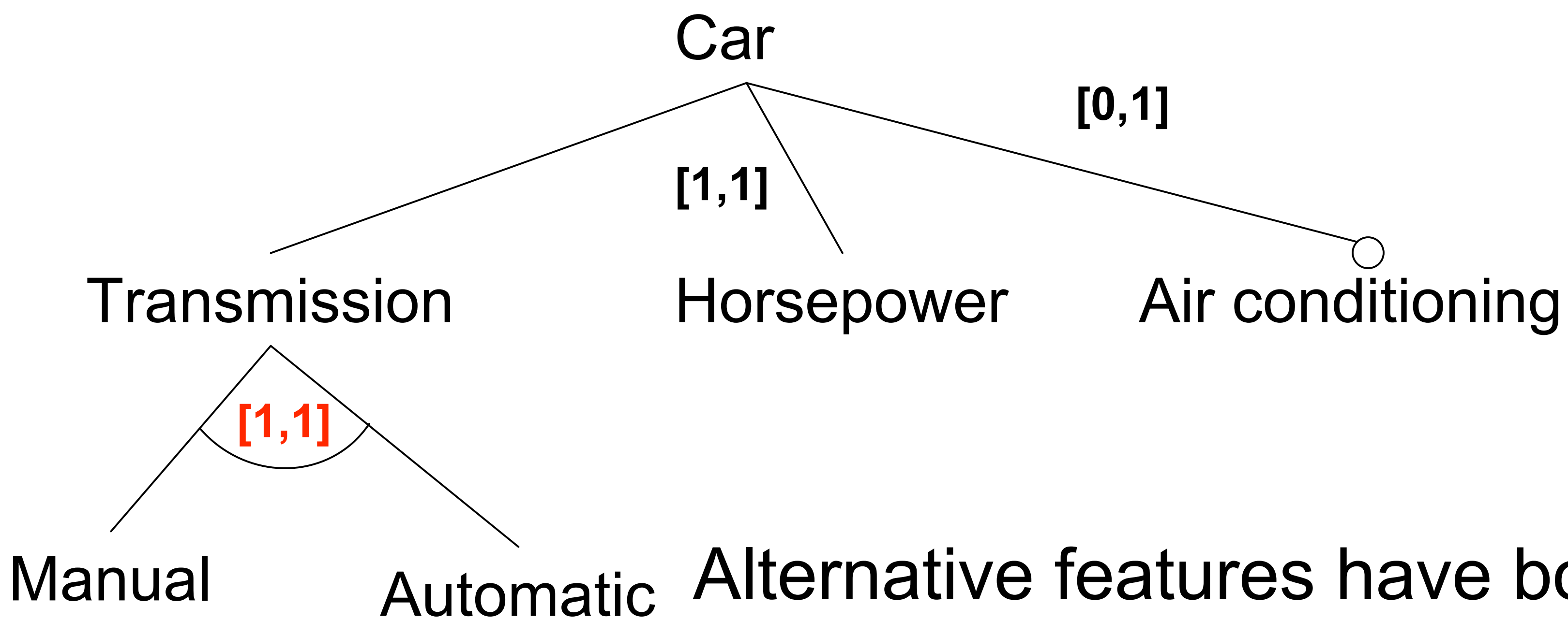
Optional features have bounds of [0,1] on the dependence

Implicit FODA Cardinalities



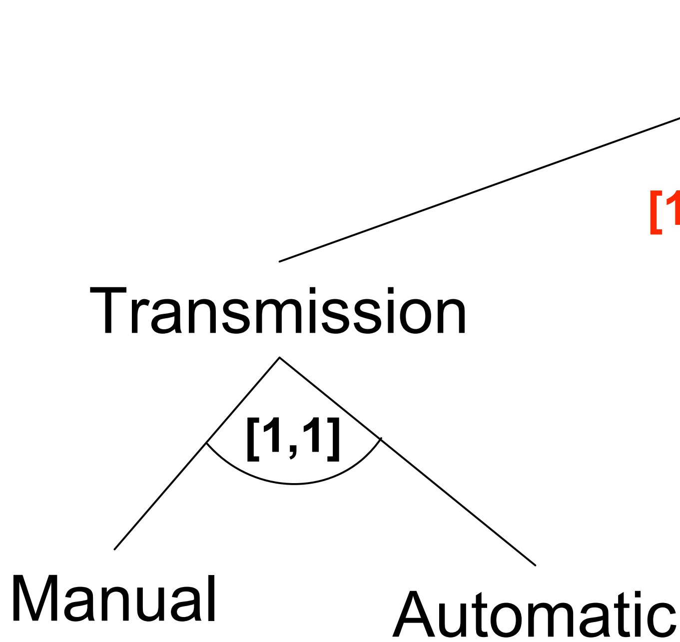
Mandatory features have bounds of [1,1] on a dependence

Implicit FODA Cardinalities



Alternative features have bounds of [1,1] on a set of dependences

Explicit Cardinalities



Provide significant expressive power over the base FODA feature models

Orthogonal Variability Model (OVM)

- Bühen, Lauenroth, Pohl (2005)
- A flat model of variability in a product line
- Basic elements:
  - Variation points
  - Variants
  - Variability dependences (with cardinalities)
  - Constraints

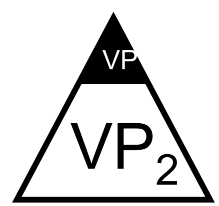
Variation Points

A set of VPs defines all of the ways a PL may vary

Not organized as a tree (ala feature models)

- hierarchy can be modeled with constraints

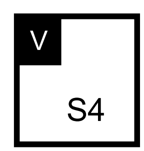
Modeled diagrammatically as



Variants

A set of variants defines the possible ways that a variation point may be realized in a PL

Variants correspond to leaves of a feature diagram

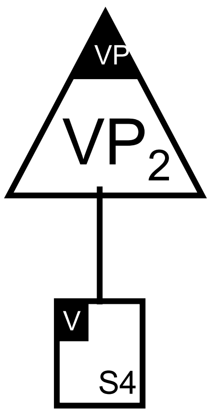
Modeled diagrammatically as 

Variability Dependences

Relate variation points to the variants that may bind to them in some product line instance

Three kinds of dependences

- Optional
- Alternative Choice
- Mandatory : a commonality depicted as

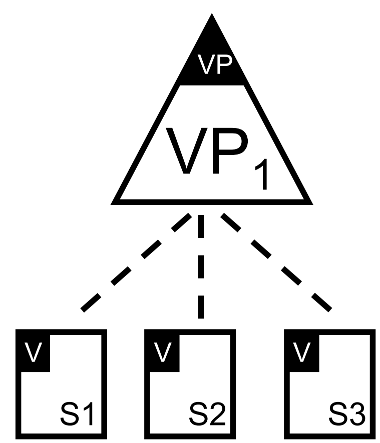


Optional Dependences

Expresses that a variants may be bound to a given variation point in a PL instance

Correspond to alternatives features in FODA

Modeled diagramatically as

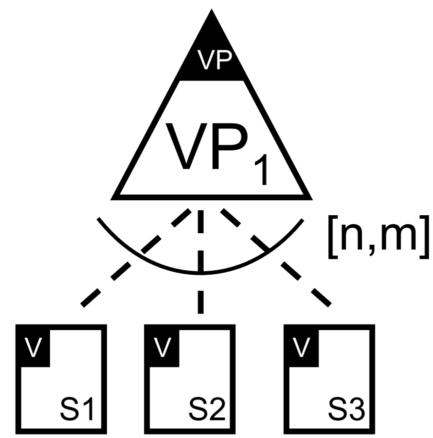


Alternative Choice Dependences

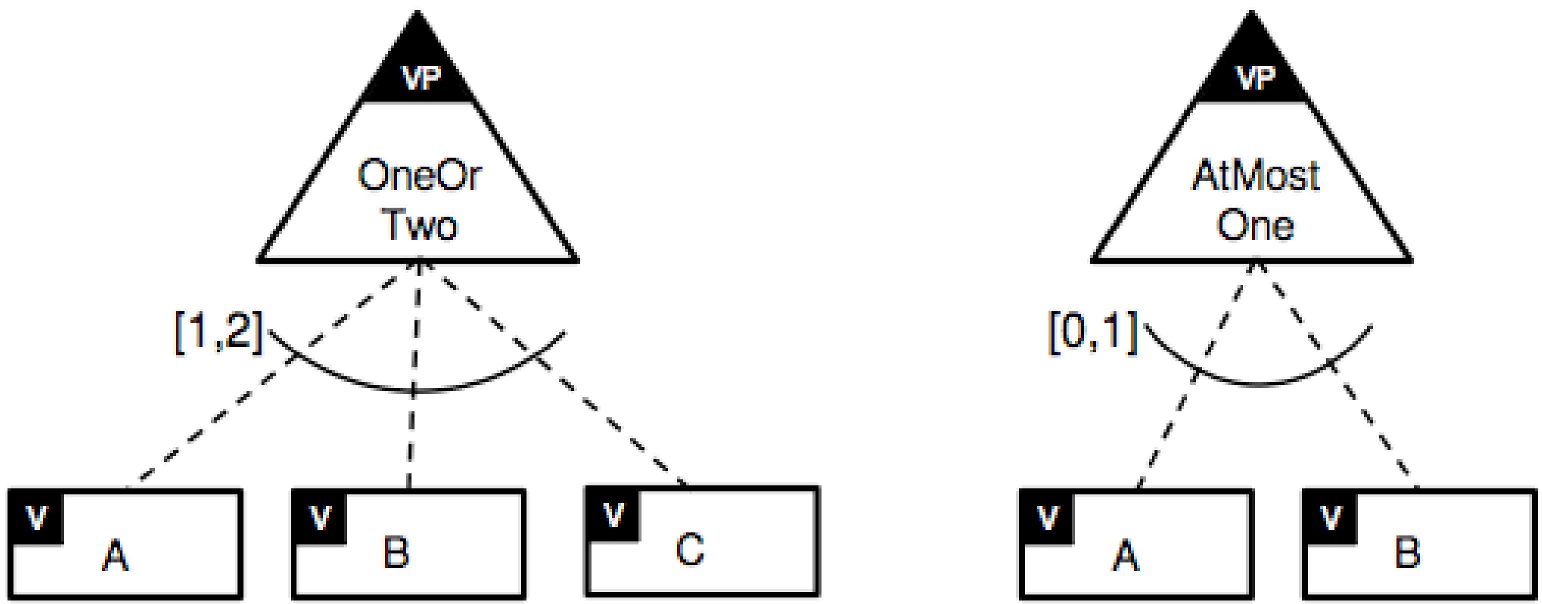
Expresses that at least n and at most m of a set of optional variants are bound to a given variation point in all product line instances

Incorporates dependence cardinalities

Modeled diagramatically as (n and m default to 1)



Alternative Choice Dependences



Constraints

Restrict the binding of dependent variants to variation points in a product line instance

There are three classes of constraints

– Variant (v_v)
– Variation Point (vp_vp)
– Variant to Variation Point (v_vp)

Within each class there can be

– requires : make allowable bindings explicit
– excludes : make unallowable bindings explicit

Variant to Variant Constraints

Restrict the binding of specific variants in an instance

NB: dependent VPs are implicit

V1 requires V2

- If V1 is in a PL instance, then V2 must be in that instance

V1 excludes V2

- If V1 is in a PL instance, then V2 cannot be in that instance

Allows for specification of feature sets

- Sets of variants that are active together

More Variant Constraints

Constraints are directed

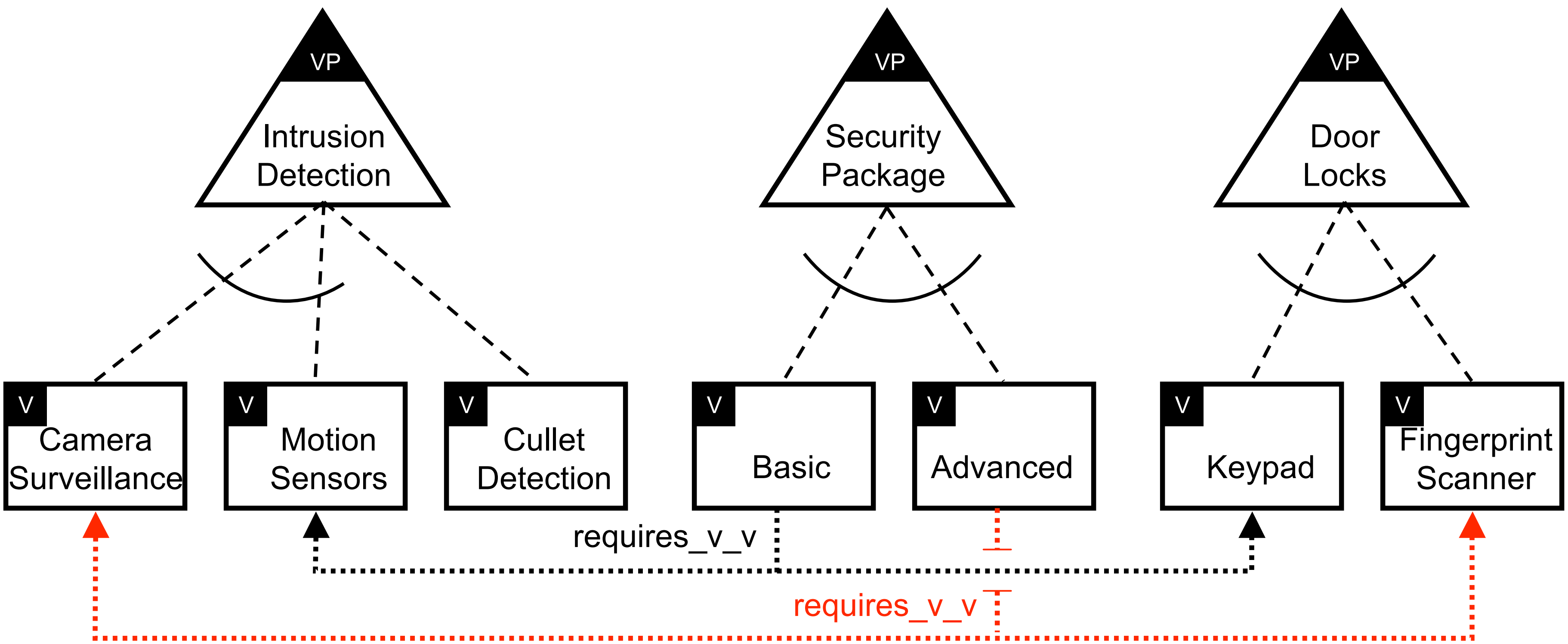
- e.g., “V1 requires V2” demands nothing of V1

Multiple constraints originating from a variant union the targets of the constraint

- e.g., V1 requires {V2,V3}

Modeled diagramatically as dashed hyper-edges

Example from Pohl 05
Part of a home security detection system

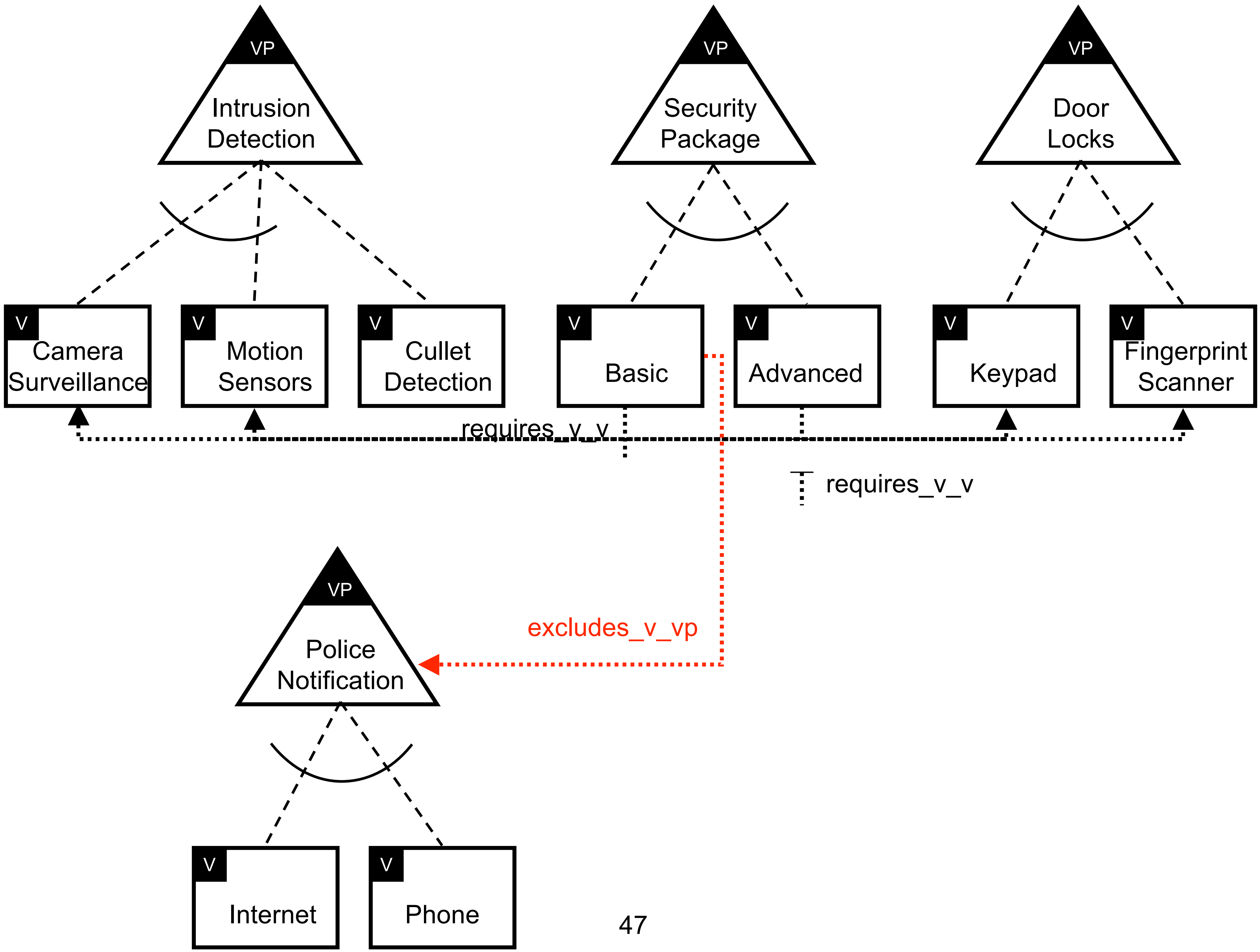


Variant to VP Constraints

Controls the inclusion of a VP based on the inclusion of a variant in a PL instance

By default, we consider all VPs in an OVM model to contribute to the description of the product line instance

In certain product lines, we may have instances in which certain VPs play no role


VP
Intrusion Detection
VP
Security Package
VP
Door Locks
V
Camera Surveillance
V
Motion Sensors
V
Cullet Detection
V
Basic
V
Advanced
V
Keypad
V
Fingerprint Scanner
requires_v_v
requires_v_v
VP
Police Notification
excludes_v_vp
V
Internet
V
Phone

VP to VP Constraints

Controls the inclusion of a VP based on the inclusion of another VP in a PL instance

Another level of generality that is useful in describing complex product lines

Can be used to hierarchies of VPs

- e.g., express hierarchical dependences between VP via requires_vp_vp

Formalizing Variability Models

Subsequent to FODA there have been a number of misinterpretations of feature models

Czarnecki observed the need for a formal definition of feature models to resolve such ambiguity

Formalization also has value in enabling

- reasoning about properties of an SPL
- application of existing V&V techniques

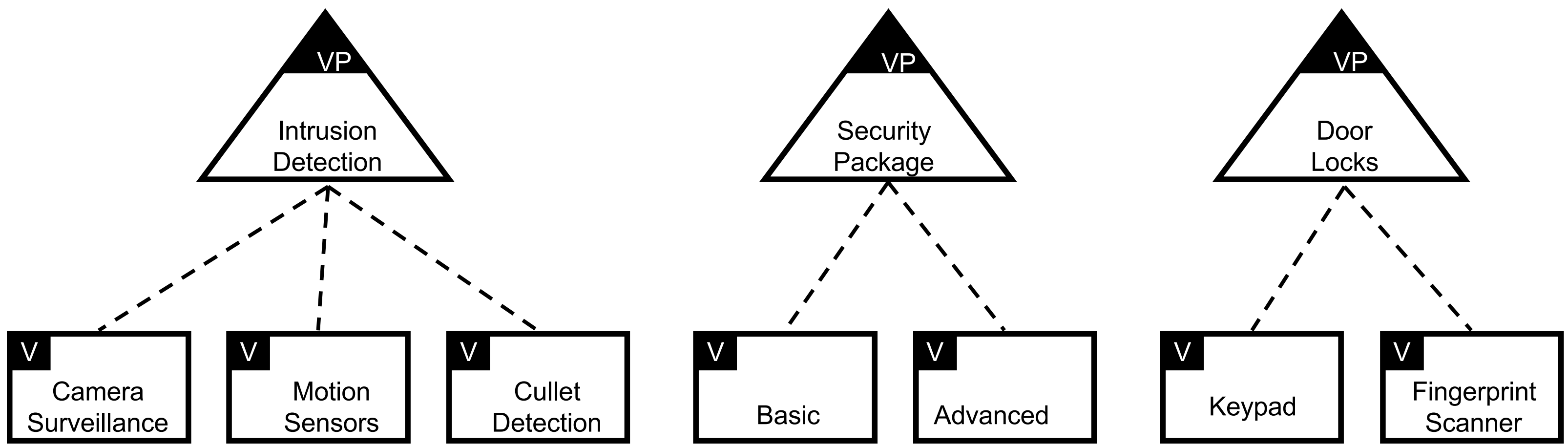
Basic Approach

Ignore mandatory dependences

Define a relational model whose tuples encode combinations of variants

Apply constraints to eliminate tuples that do not correspond to feasible instances of the PL

Resulting relation defines the extent of the PL model

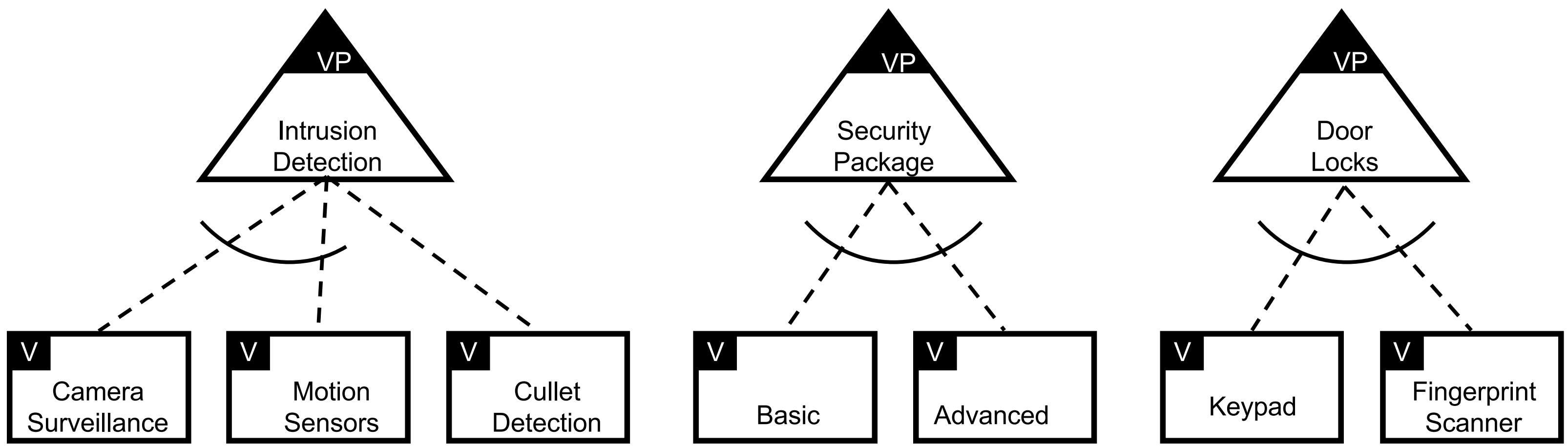


Optional semantics yields

8*4*4 = 128 tuples

Must account for any subset of the optional variants

including none

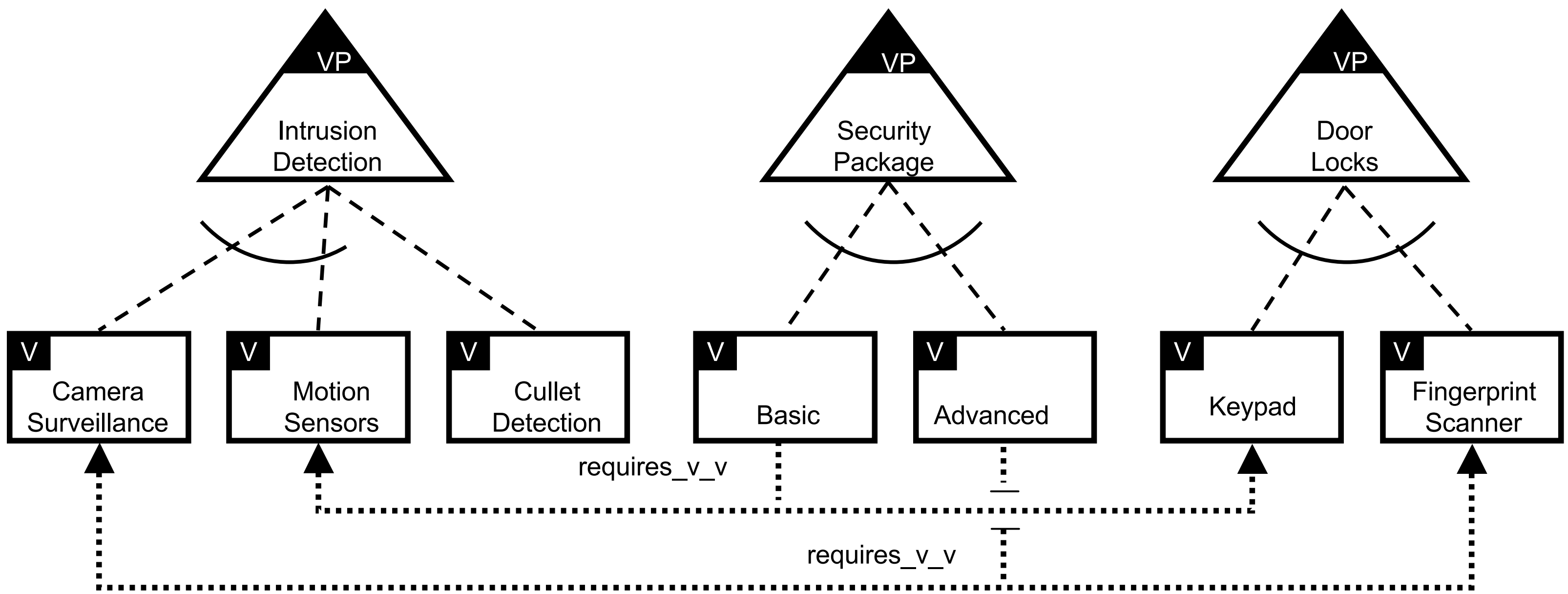


Associative choice semantics yields

2*2*2*2 = 16 tuples

Select exactly one variant from each choice group

Remaining options treated as before



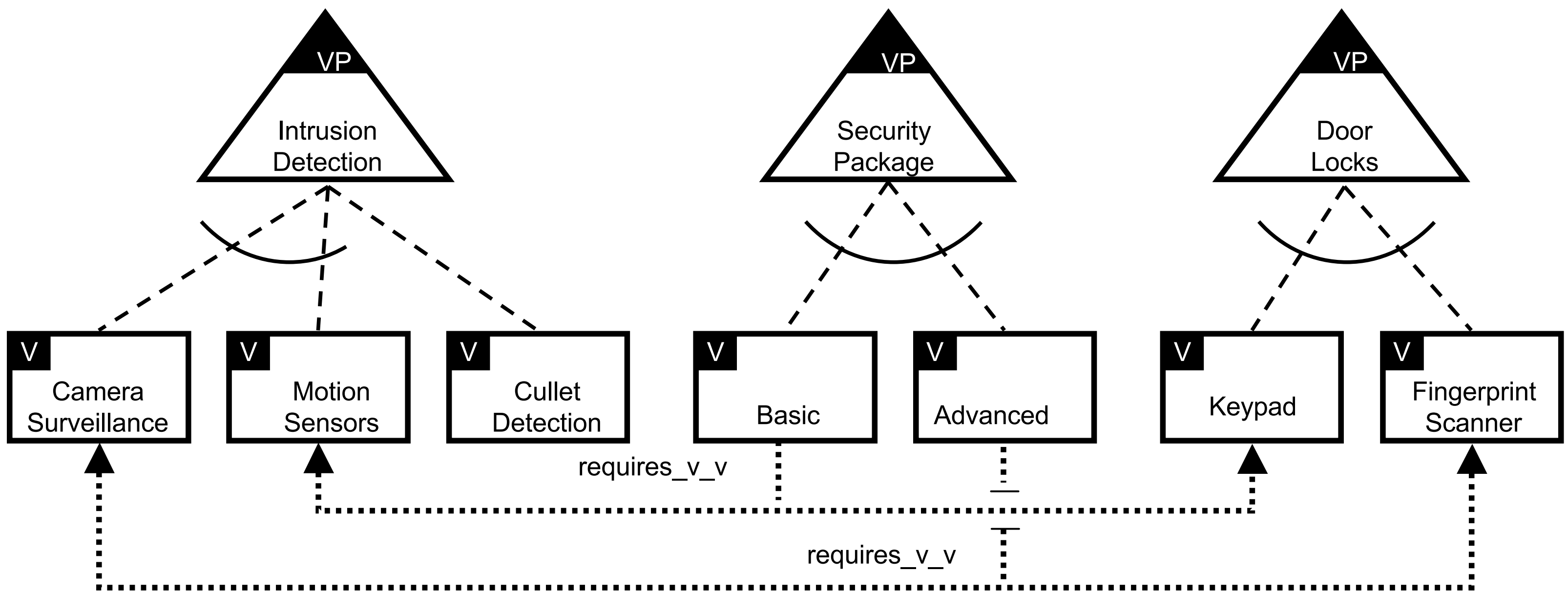
Constraints reduce the set of tuples

Basic requires Motion Sensors

eliminates tuples with Basic and Camera Surveillance

Basic requires Keypad

eliminates tuples with Basic and Fingerprint Scanner



Values	Factor			
	Intrusion Detection A	***Intrusion Detection B***	***Security Package***	***Door Locks***
	Camera Surv.	Cullet Det.	Basic	Keypad
	Camera Surv.	None	Basic	Keypad
	Motion Sen.	Cullet Det.	Advanced	Finger. Scanner
	Motion Sen.	None	Advanced	Finger. Scanner

Simple Relational Models

Domain: finite set of values - *D*

Relation: a subset of the Cartesian product of some number of domains.

Relation over *k* domains - $\prod_{i=1}^{k} D_i$

Elements of a relation are tuples

Simple Relational Models

With k factors we have a k-tuple $(v_1, v_2, \ldots, v_k)$ where $v_i \in D_i$

To extract a value for a factor, i, from a tuple, $t=(v_1,\ldots v_k)$, use a projection function

$$\pi(t,i) = v_i$$

where $1 \le i \le k$.

Basic OVM Mapping

Variation point: modeled by a set of factors denoted f(vp) for some variation point vp

Variants: modeled as values

Variability dependencies: relate a set of variants to a variation point (defines the domain)

Basic OVM Mapping

Mandatory dependences

ignore since these do not vary

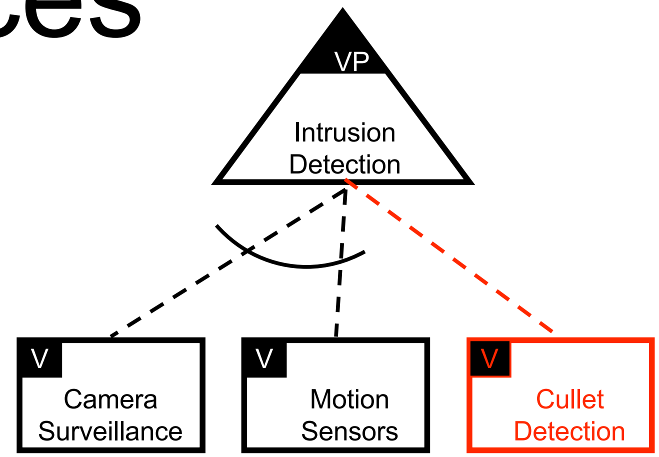
Optional dependences

introduce multiple factors for a variation point to allow a variation point to be related to a set of variants

Associative choice dependences

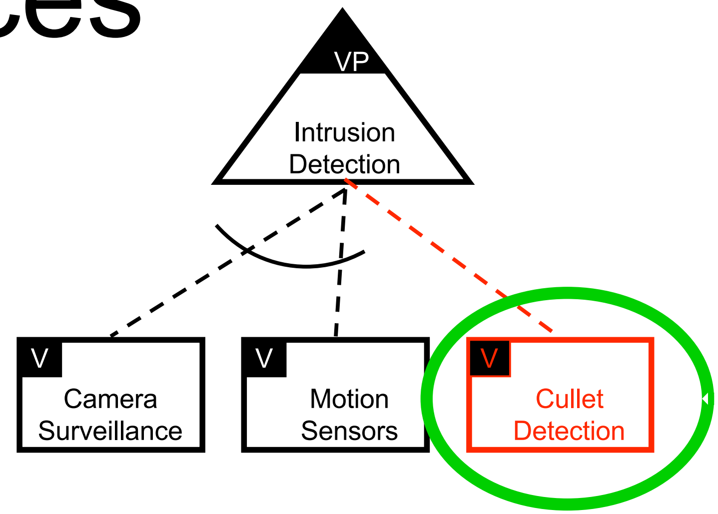
more complex

Optional Dependences



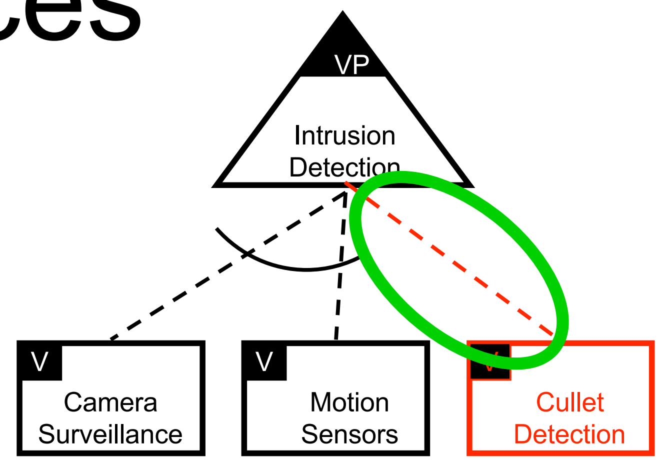
Values	Factor			
	Intrusion Detection A	***Intrusion Detection B***	***Security Package***	***Door Locks***
	Camera Surv.	Cullet Det.	Basic	Keypad
	Camera Surv.	None	Basic	Keypad
	Motion Sen.	Cullet Det.	Advanced	Finger. Scanner
	Motion Sen.	None	Advanced	Finger. Scanner

Optional Dependences



Values	Factor			
	Intrusion Detection A	***Intrusion Detection B***	***Security Package***	***Door Locks***
	Camera Surv.	Cullet Det.	Basic	Keypad
	Camera Surv.	None	Basic	Keypad
	Motion Sen.	Cullet Det.	Advanced	Finger. Scanner
	Motion Sen.	None	Advanced	Finger. Scanner

Optional Dependences



Values	Factor			
	Intrusion Detection A	***Intrusion Detection B***	***Security Package***	***Door Locks***
	Camera Surv.	Cullet Det.	Basic	Keypad
	Camera Surv.	None	Basic	Keypad
	Motion Sen.	Cullet Det.	Advanced	Finger. Scanner
	Motion Sen.	None	Advanced	Finger. Scanner

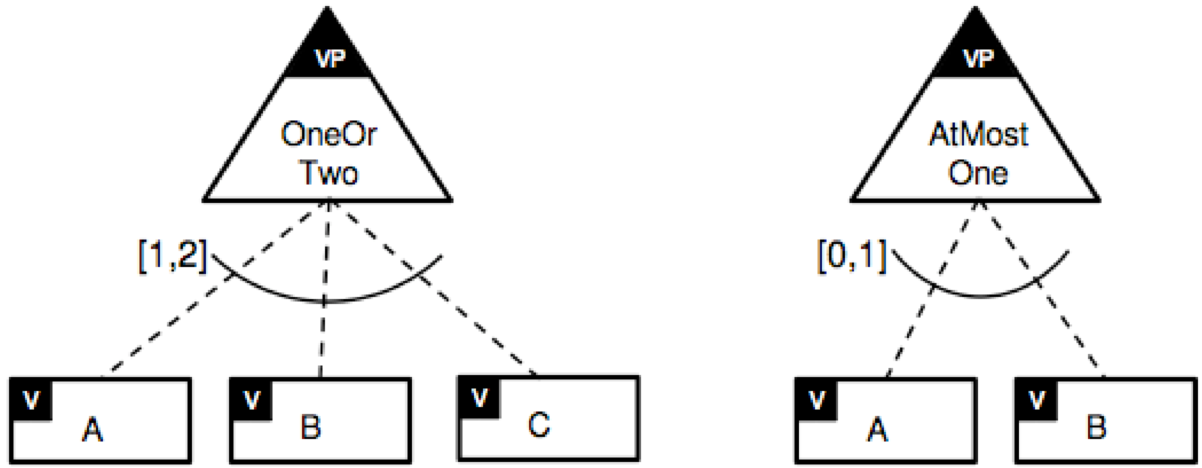
Alternative Choice Dependencies

Given an alternative choice with bounds [i,j]

Introduce *i* factors for the variation point with domain defined by the exact set of dependent variant values

Introduce *j-i* factors for the variation point with a domain defined by the set of variant values and Ø (the empty value)

Alternative Choice Dependencies



$OneOrTwo_1$: {A, B, C}

AtMostOne : {A,B, Ø}

$OneOrTwo_2$: {A,B,C,Ø}

f(OneOrTwo) = {$OneOrTwo_1$,$OneOrTwo_2$}

Alternative Choice Dependence

OVM allows a variant to be bound at most once

We could produce a tuple, t, for OneOrTwo such that $\pi(t, OneOrTwo_1) = \pi(t, OneOrTwo_2) = A$

To avoid this add inequality constraints between all pairs of factors f(vp)

Basic OVM Mapping Size

In the worst case where all dependences are optional an OVM model with *k* variants gives rise to a relational model with *k* factors

However, alternative choices with default [1,1] bounds seem very common so we expect the number to be closer to the number of variation points since a single factor is needed for a VP

Mapping OVM Constraints

An unconstrained OVM model is:

$$U = \prod_{vp \in OVM} \prod_{f \in f(vp)} D_f$$

tuples of the unconstrained model *over approximate* the set of feasible product line instances

Constraints

Strategy:

- Define sub-relations of U that are consistent with each constraint
- Intersect resulting constraints

Example (non-$\emptyset$ Inequality Constraint):

$$I(i,j) = \{t \mid t \in U \wedge (\pi(t,i) \neq \emptyset \Rightarrow \pi(t,i) \neq \pi(t,j))\}$$

Cumulative Inequality Constraints

For a variation point, vp:

$$I(vp) = \bigcap_{i \in f(vp),\, j \in f(vp) - \{i\}} I(i, j)$$

For an OVM model:

$$I = \bigcap_{vp \in OVM} I(vp)$$

Explicit OVM Constraints

A requires_v_v constraint on factor *i*, with variant *v*, and factor *j*, with variant *w*

$$\begin{aligned} R(i,v,j,w) \quad = \quad & \{t \mid t \in U \wedge (\exists_{f \in f(i)} : \pi(t,f) = v) \wedge \\ & (\exists_{f \in f(j)} : \pi(t,f) = w)\} \cup \\ & \{t \mid t \in U \wedge (\forall_{f \in f(i)} : \pi(t,f) \neq v)\} \end{aligned}$$

Explicit OVM Constraints

A requires_v_v constraint on VP *i*, with variant *v*, and VP *j*, with variant *w*

$$R(i, v, j, w) = \{t \mid t \in U \wedge (\exists_{f \in f(i)} : \pi(t, f) = v) \wedge (\exists_{f \in f(j)} : \pi(t, f) = w)\} \cup \{t \mid t \in U \wedge (\forall_{f \in f(i)} : \pi(t, f) \neq v)\}$$

When VP *i* has value v, then we require something of the value of VP *j*

Explicit OVM Constraints

A requires_v_v constraint on VP *i*, with variant *v*, and VP *j*, with variant *w*

$$R(i, v, j, w) \quad = \quad \{t \mid t \in U \wedge (\exists_{f \in f(i)} : \pi(t, f) = v) \wedge (\exists_{f \in f(j)} : \pi(t, f) = w)\} \cup \{t \mid t \in U \wedge (\forall_{f \in f(i)} : \pi(t, f) \neq v)\}$$

When VP *i* has a different value, then we make no requirement of the value of VP *j*

Combining Relational Models

All of our constraints are sub-relations

We can combine them through intersection with U

A constrained OVM model is

$$U \bigcap I \bigcap R(\ldots) \bigcap E(\ldots)$$

Limitations

Czarnecki's approach allows for

- recursive feature diagrams
- multiple instances of a variant for a VP

Batory has suggested propositional constraints

References

Kang, K., Cohen, S., Hess, J., Nowak, W., and Peterson, S., Feature-oriented Domain Analysis (FODA) Feasibility Study, Technical Report CMU/SEI-90TR-21, Software Engineering Institute, Carnegie Mellon, Pittsburgh, PA (1990)

Czarnecki, K., Helsen, S., and Eisenecker, U., Staged Configuration Using Feature Models, Software Product Line Conference, LNCS 3154 (2004)

Czarnecki, K., Helsen, S., and Eisenecker, U., Formalizing Cardinality-based Feature Models and their Specialization, Software Process Improvement and Practice (2005)

Czarnecki, K., and Kim, C.H.P., Cardinality-based Feature Modeling and Constraints : A Progress Report, OOPSLA'05 Workshop on Software Factories (2005)

Buhne, S., Lauenroth, K., and Pohl, K., Modelling Requirements Variability across Product Lines, IEEE Symposium on Requirements Engineering (2005)

Pohl, K., Bockle, G., and van der Linden, F., Software Product Line Engineering : Foundations, Principles, and Techniques, Springer (2005)

Batory, D., Feature Models, Grammars, and Propositional Formulas, Software Product Line Conference, LNCS 3714 (2005)