An introduction to Software Product Lines

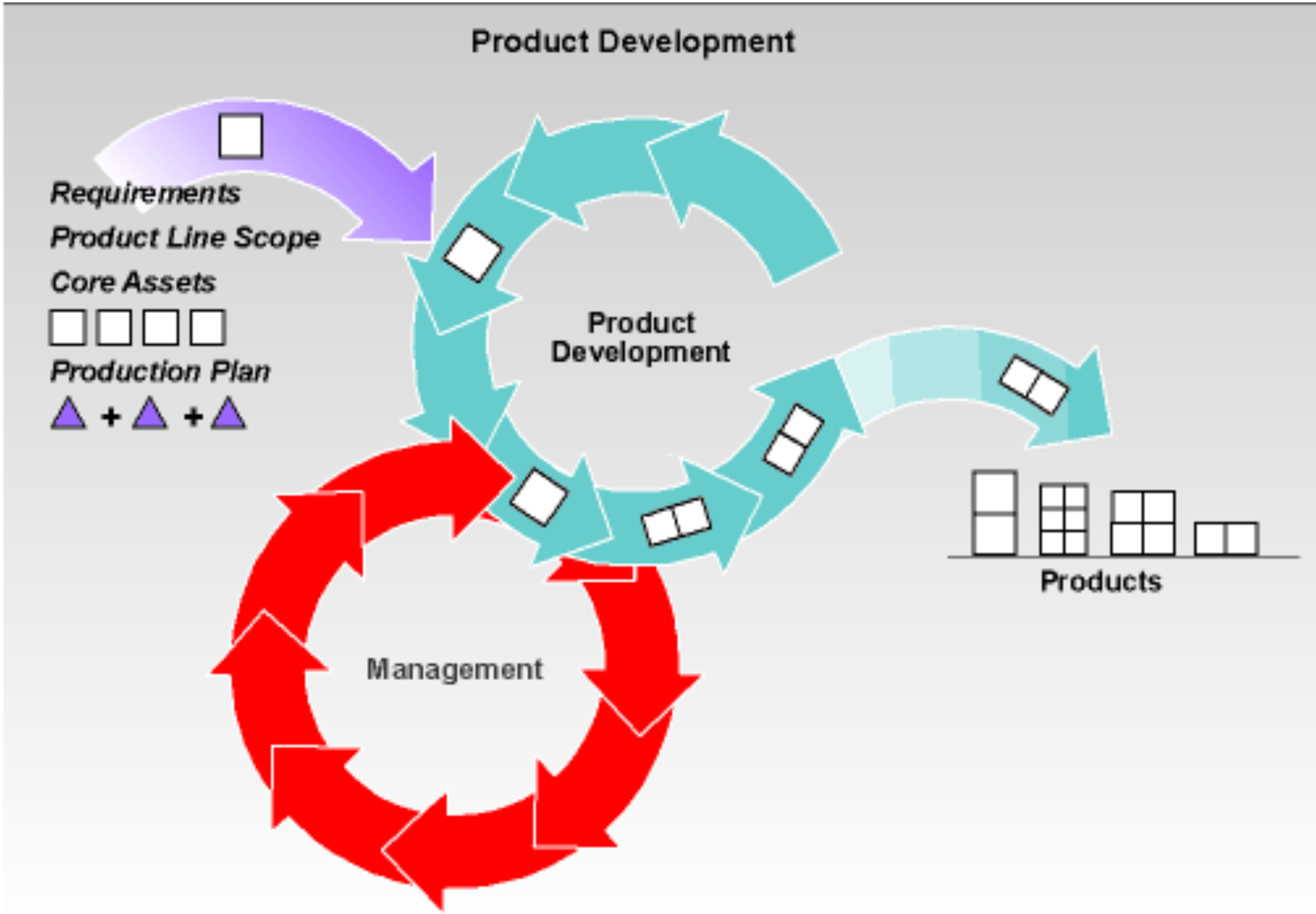
Maurice ter Beek
ISTI-CNR
maurice.terbeek@isti.cnr.it

These slides are partly based on slides by Klaus Pohl, Sven Apel and Slinger Jansen

Outline

- (Software) Product Line Engineering
- Product variability and features
- Variability modelling and analysis

Product line engineering in theory…



…and in practice

Software Product Line Engineering

- Developing a family of products from a common reference model (usually in terms of features) and mass customisation (to serve different markets)

Two main differences with classical software engineering

- Two distinct development processes
  - Domain engineering: develop reusable domain artefacts
  - Application engineering: develop individual products by reusing domain artefacts
- Variability
  - Common features that are part of all products
  - Variable features that can be selected for a product

Variability in requirements

R12: The navigation system must allow the user to make inputs using a control panel or by voice entry

- R12 comprises the following three realisations
  1. A navigation system that allows the user to make inputs only via the control panel
  2. A navigation system that allows the user to make inputs only via voice entry
  3. A navigation system that allows the user to make inputs only via the control panel and by voice entry
- Conjunction is a logical “or” or an exclusive “or”?
- Is only one system asked for, or two or three different systems?

Variation points and variants

- A variation point represents an aspect of a product family that varies among the different products

  R12: “input modality of the user interface”

- A variant represents a specific configuration (i.e. an incarnation) of a variable aspect that a product in a product family can have

  R12: “input via control panel”, “input via voice entry”

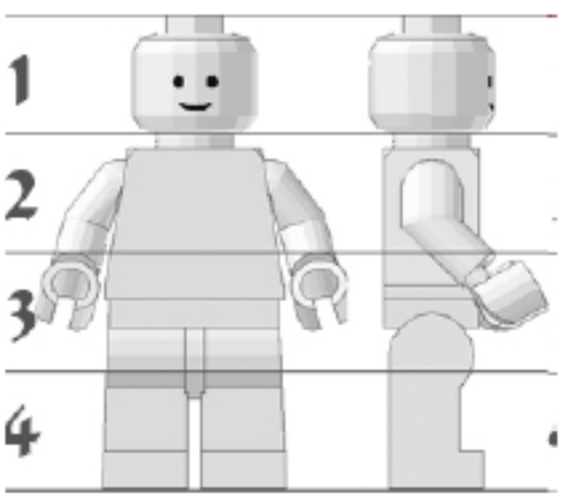
Variability: prerequisite for success

- Domain engineering: explicit documentation of variability supports the identification of possible variable aspects and fosters explicit decisions about which aspects shall be variable in the product family (variation points) and which options shall exist for each variable aspect (variants);

  it also supports engineers, architects, designers and testers in realising the defined variation points and variants
- Application engineering: explicit documentation of variability in terms of variation points and variants supports system development by making explicit the necessary decisions and decision options

Features

- What is a feature?
  - End-user visible behaviour or property of a system…
  - …that may be optional and/or may have alternatives
- Features represent commonalities and variabilities of (software) systems



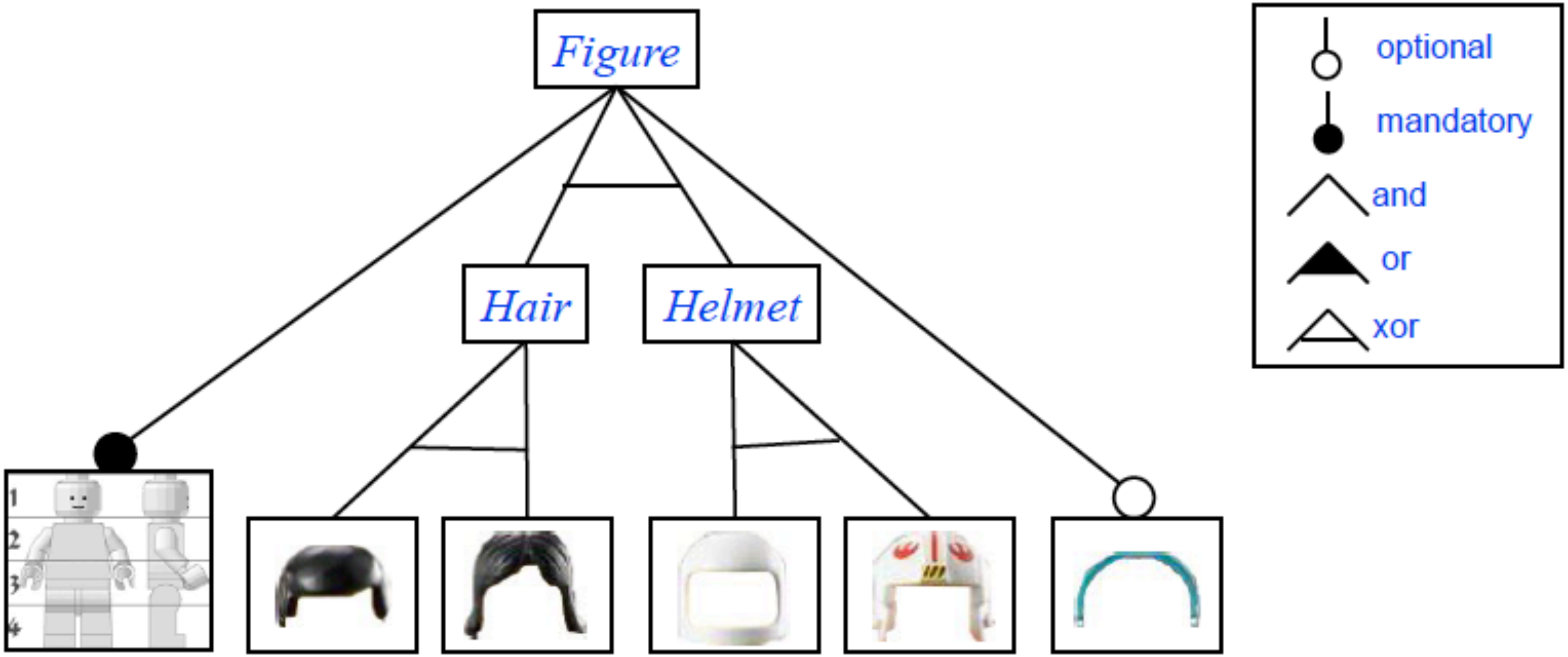
What's in a *feature*?

Reference	Definition
Kang *et al.* [3]	*"a prominent or distinctive user-visible aspect, quality or characteristic of a software system or systems"*
Kang *et al.* [8]	*"distinctively identifiable functional abstractions that must be implemented, tested, delivered, and maintained"*
Eisenecker and Czarnecki [6]	*"anything users or client programs might want to control about a concept"*
Bosch *et al.* [9]	*"A logical unit of behaviour specified by a set of functional and non-functional requirements."*
Chen *et al.* [10]	*"a product characteristic from user or customer views, which essentially consists of a cohesive set of individual requirements"*
Batory [11]	*"an elaboration or augmentation of an entity(s) that introduces a new service, capability or relationship"*
Batory *et al.* [12]	*"an increment in product functionality"*
Apel *et al.* [13]	*"a structure that extends and modifies the structure of a given program in order to satisfy a stakeholder's requirement, to implement and encapsulate a design decision, and to offer a configuration option."*

(Classen *et al.* @ FASE 2008)

Feature selection

Typically, only a subset of feature combinations is valid



$Visor \Rightarrow Helmet$

Products and product lines

- Product ⇔ valid feature combination (configuration)

- Product line ⇔ set of valid feature combinations of a domain

...

Features and software

- Main concept in SPLE
  - Easy to use in informal models
  - Easily converts into business: product sales
  - Easily converts into product design: variability
  - Enables reuse of features
- In telecommunication, features became popular in the 1960s with the advent of computer-controlled telephone switches; telecommunication software has been conceived in terms of features ever since

Variability models: explicit decisions

- Compact representations of all products of a product family in terms of their features
- Feature diagram/model: hierarchical tree structure, with the family as its root, features as its nodes and possibly further cross-tree constraints (Kang *et al*.'90)
- Common Variability Language (CVL): OMG's effort to standardise variability modelling as a separate and generic language to define variability on base models
- Orthogonal Variability Model (Pohl *et al*. '05), *etc., etc.*

Advantages of dedicated conceptual model

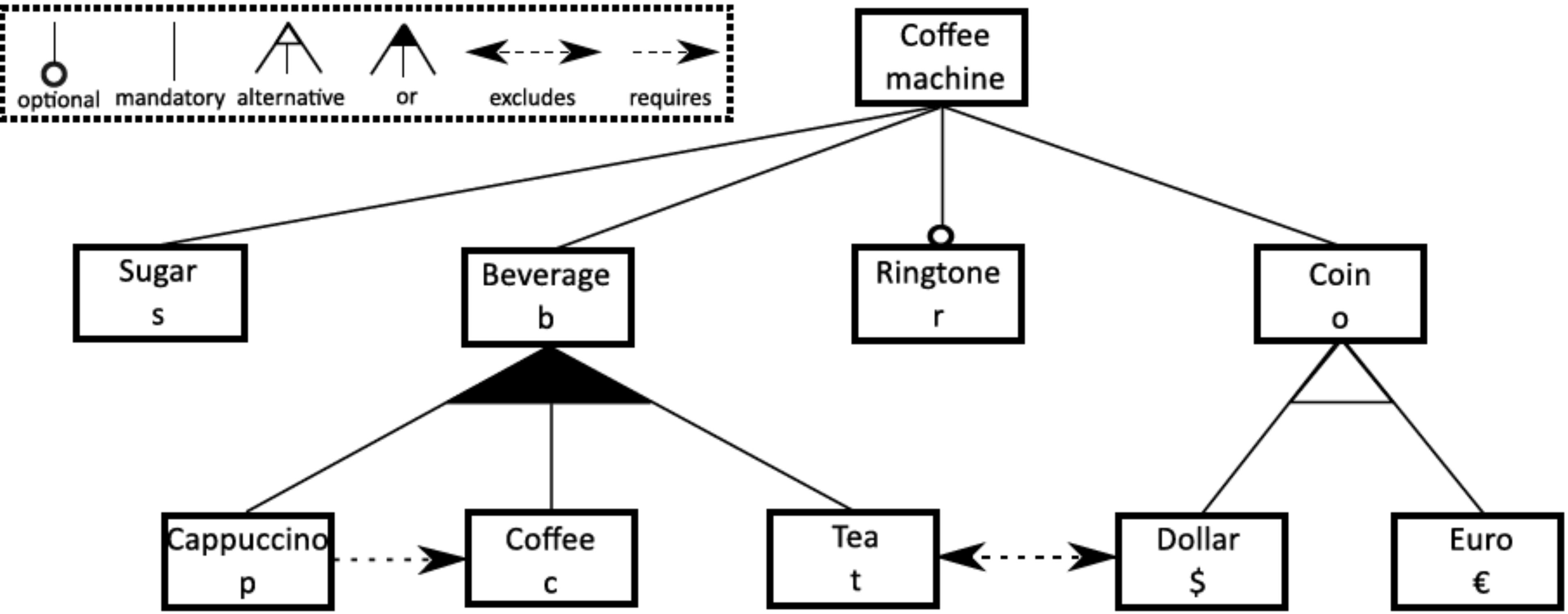
- Improved communication with different stakeholders (e.g. communicate to customers which variants can be selected at which variation points)
- Transparent decisions i.e. the originator of a variation point is forced to state the rationale for introducing variability in a specific domain artefact
- Relationships between requirements and variants become traceable (e.g. stakeholders can document which requirements, design, implementation and test artefacts are influenced by a variant)

Requirements coffee machine family

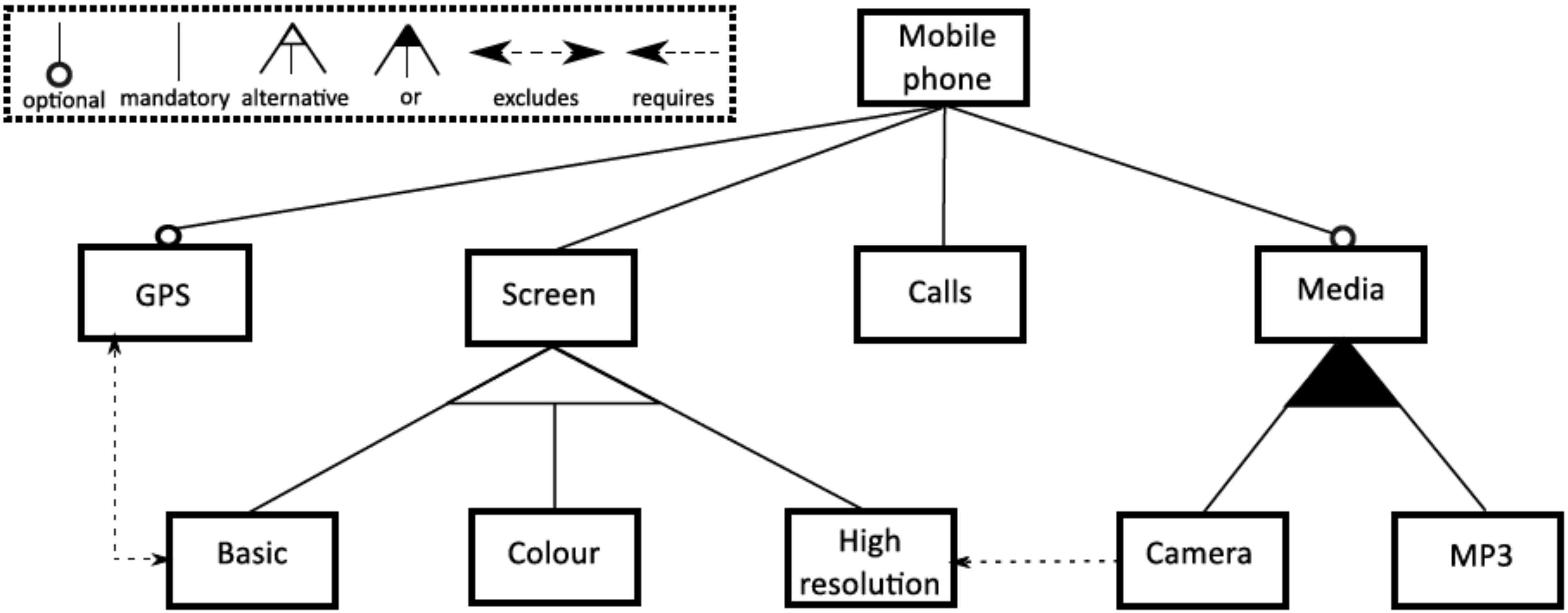
1. Initially, a coin must be inserted: either a €, exclusively in case of coffee machines for the European market, or a $, exclusively in case of coffee machines for the Canadian market (alternative features)
2. After having inserted a coin, the user has to be offered the option to choose whether or not (s)he wants sugar in her/his beverage, after which (s)he has to be offered to select a beverage (mandatory features)
3. The choice of beverages offered by a coffee machine may vary (the options being cappuccino, coffee and tea), but every coffee machine must offer at least one beverage (or relation among features);

   furthermore, whenever a coffee machine offers cappuccino, then it must offer coffee as well, while tea may only be offered by coffee machines for the European market (requires and excludes relations among features)
4. Optionally, a ringtone may be rung after the coffee machine has delivered the chosen beverage (optional feature)
5. As soon as the user has taken her/his beverage, the coffee machine must return in its idle state

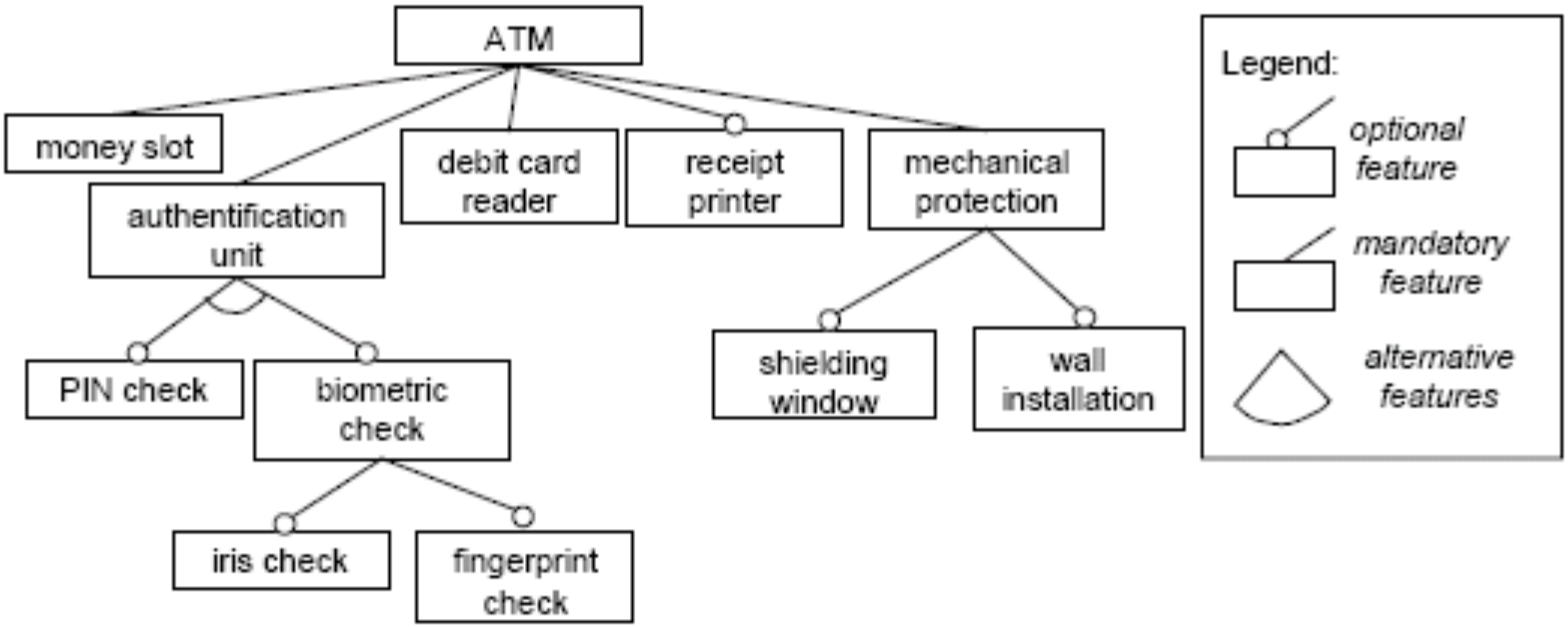
Coffee machine feature diagram



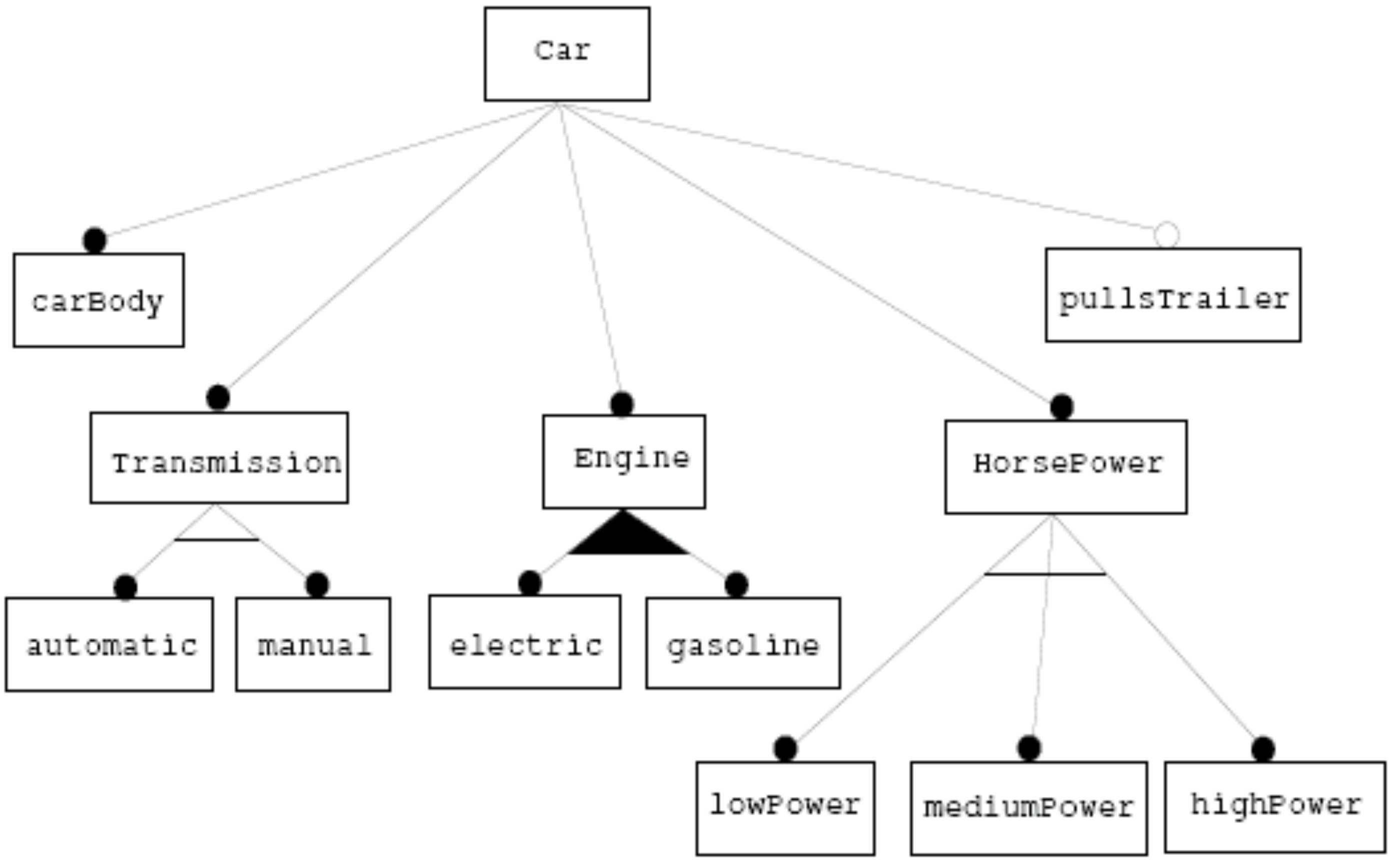
Mobile phone feature diagram



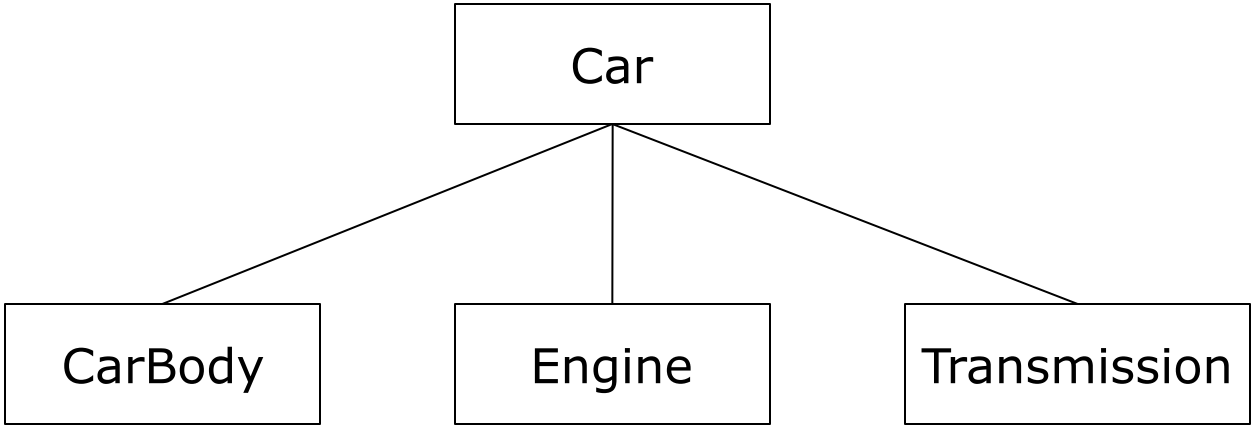
ATM feature diagram



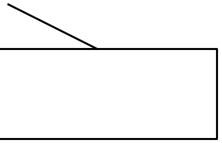
Automotive feature diagram



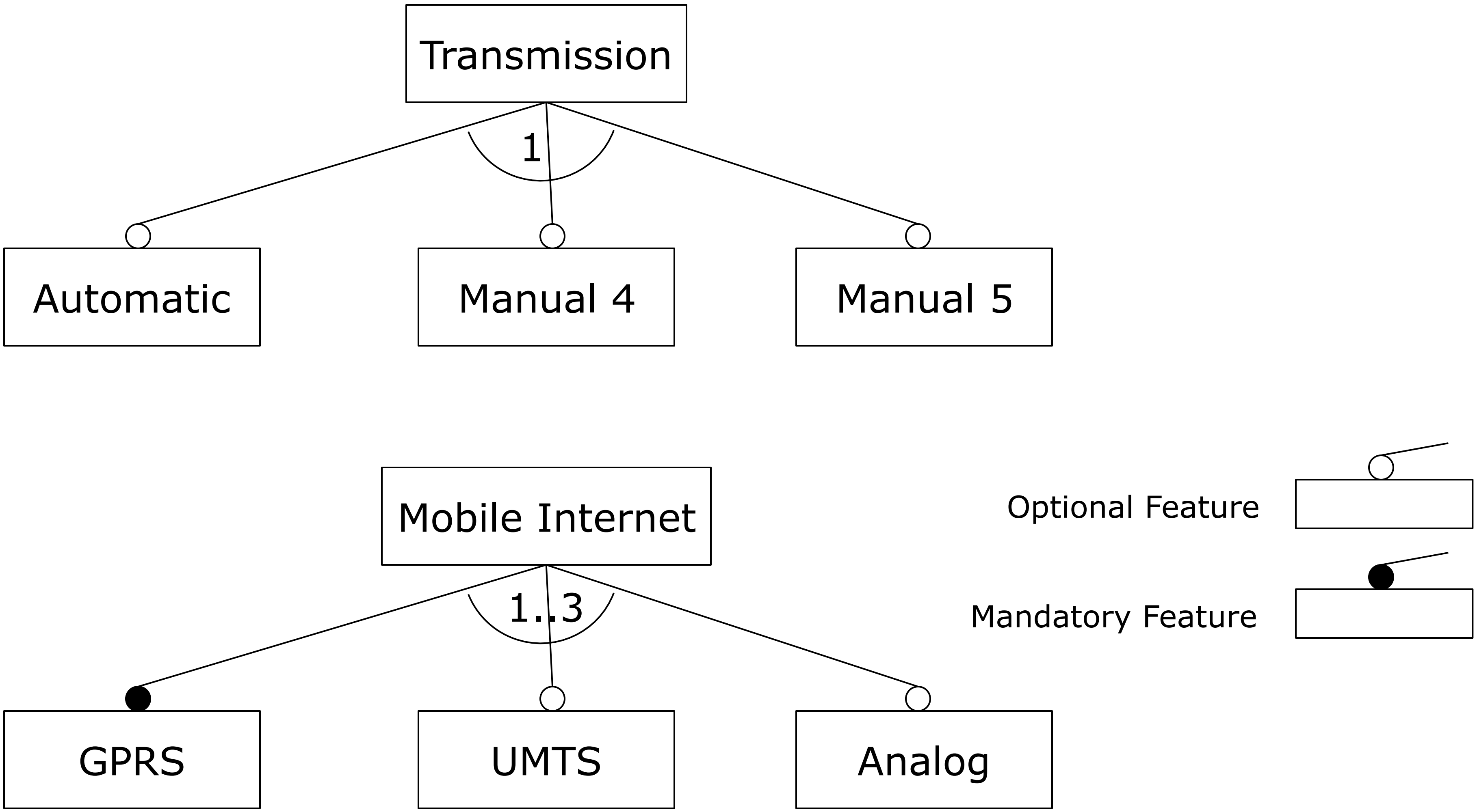
Draw up your own: Composition



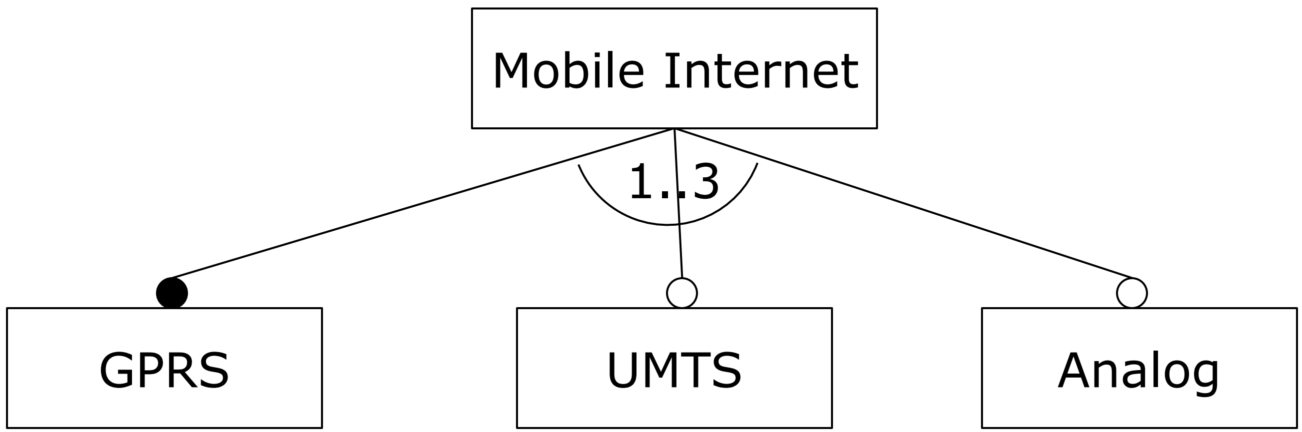
Mandatory Feature



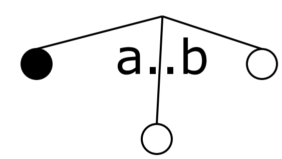
Draw up your own: Optionality (1/2)

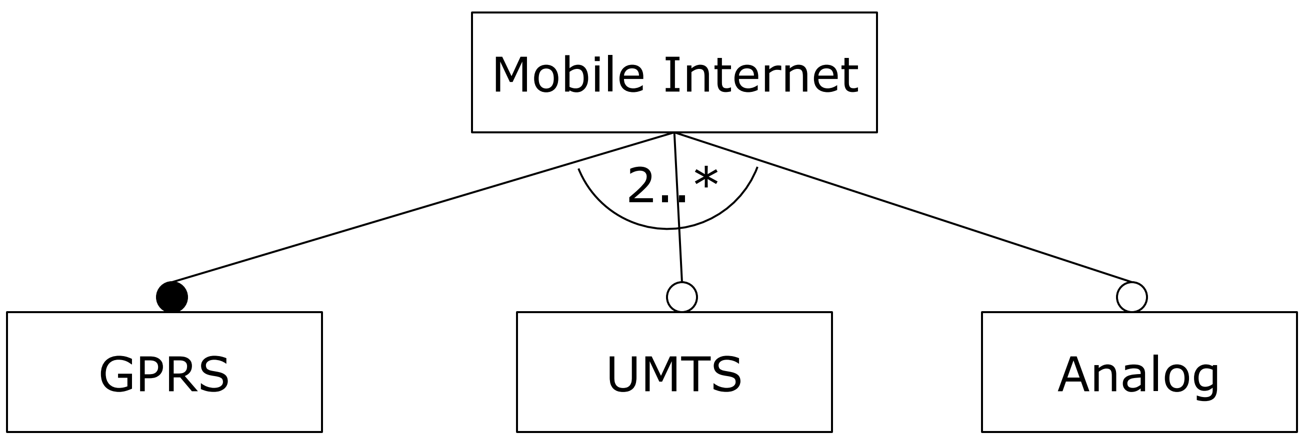


Draw up your own: Optionality (2/2)

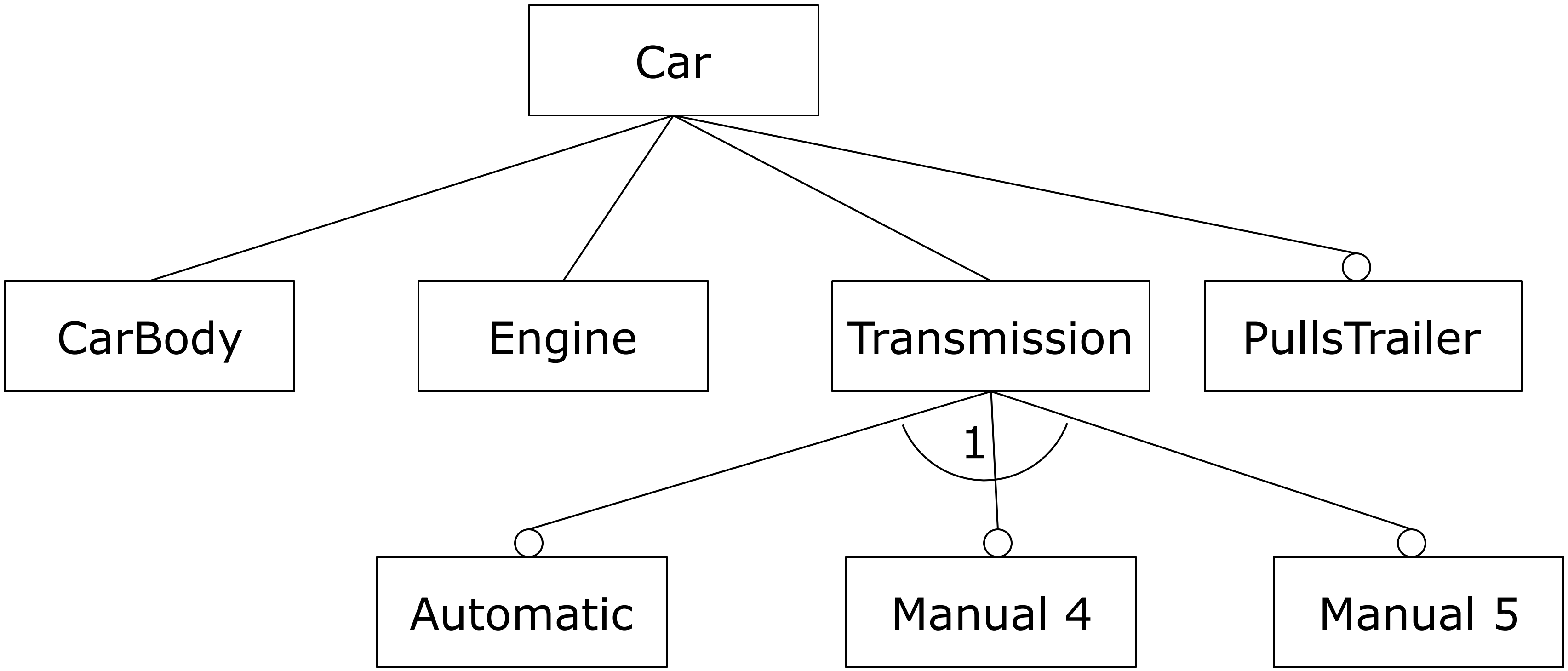


Pick a out of b features



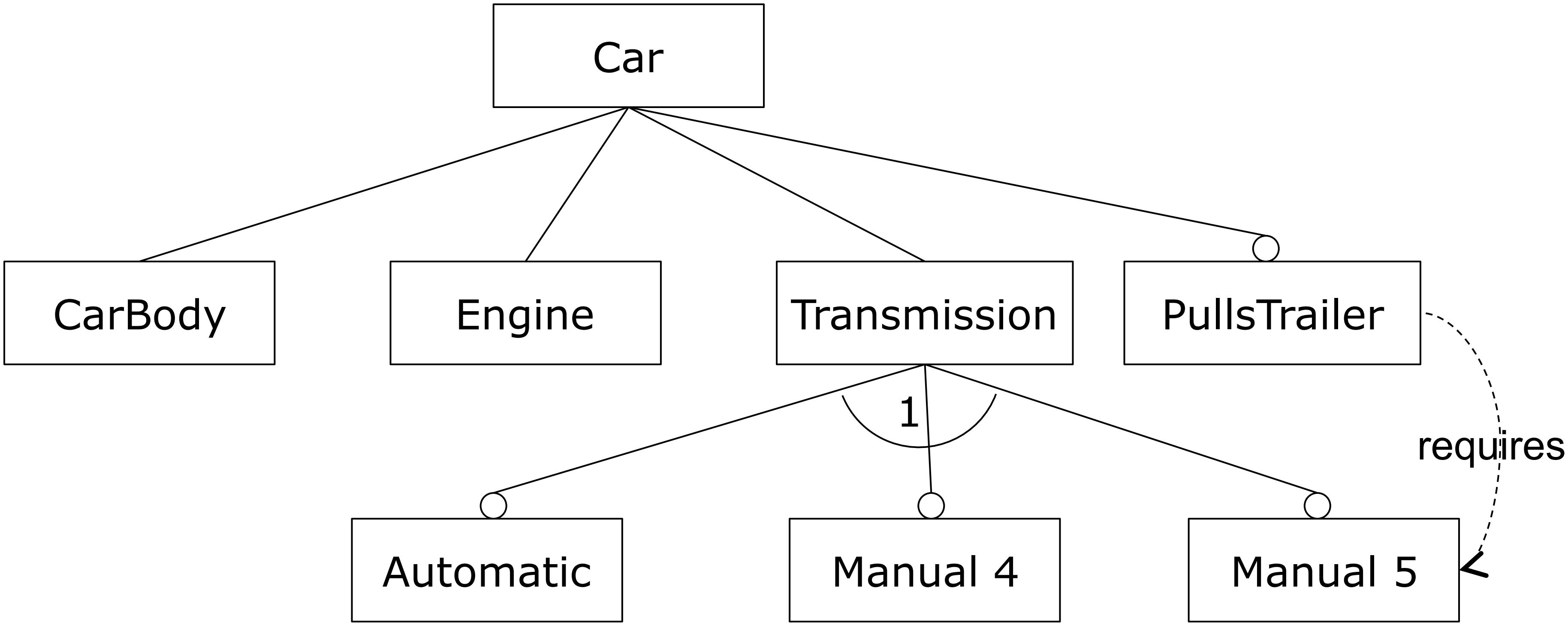


Cross-tree constraints: excludes & requires



e.g. “PullsTrailer EXCLUDES Manual 4” or “PullsTrailer REQUIRES Manual 5”

How many products?

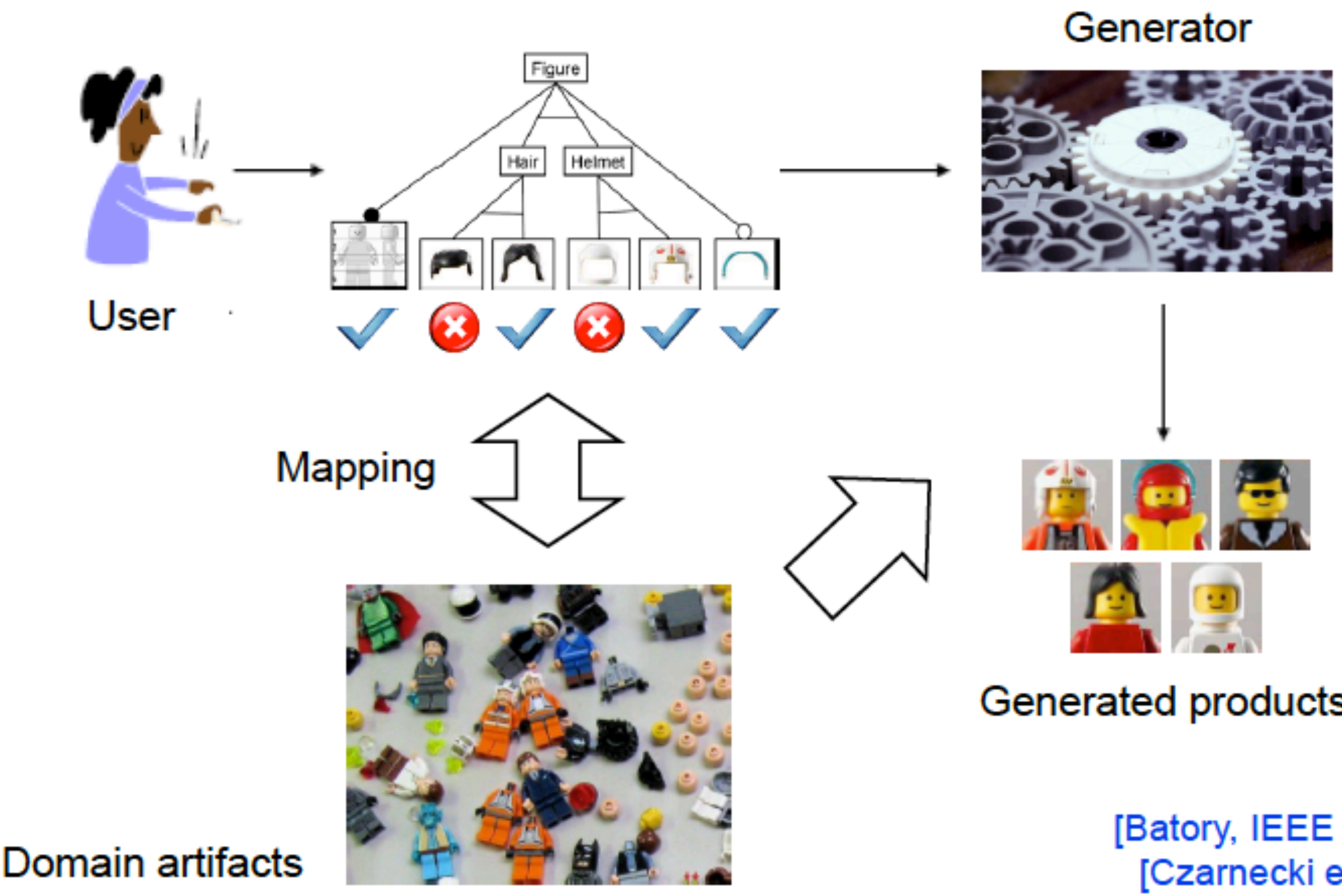


Transmission contributes one of three options, multiplier = 3
Car contributes two options, multiplier = 2, but…
…actually only works in one case; thus: 4 different products

Applications of feature diagrams

- Communicate with stakeholders
- Identify objects for reuse
- Identify objects for sales opportunities
- Identify cross-cutting concerns
- Software composition and deployment
- ...

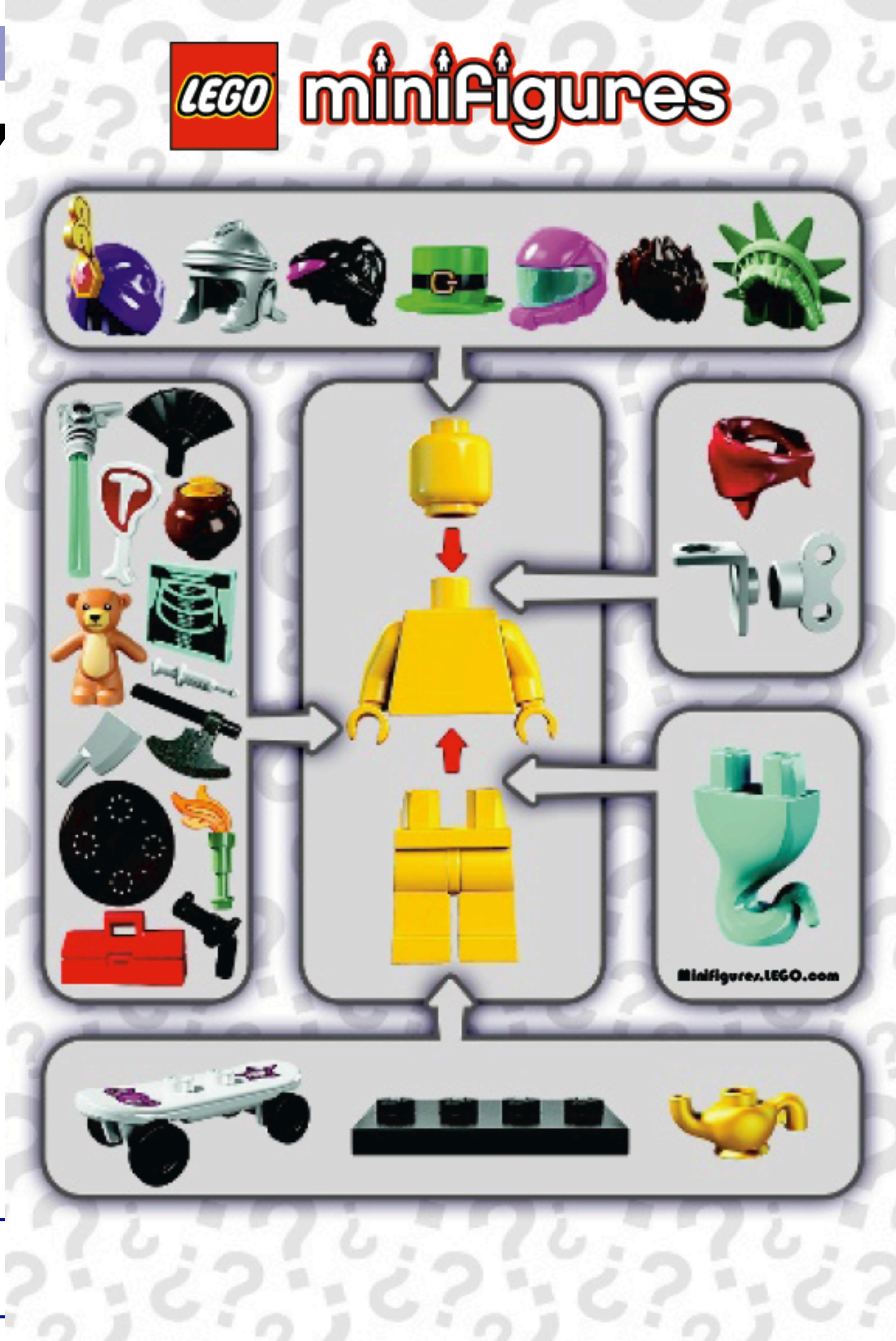
… Automated product derivation



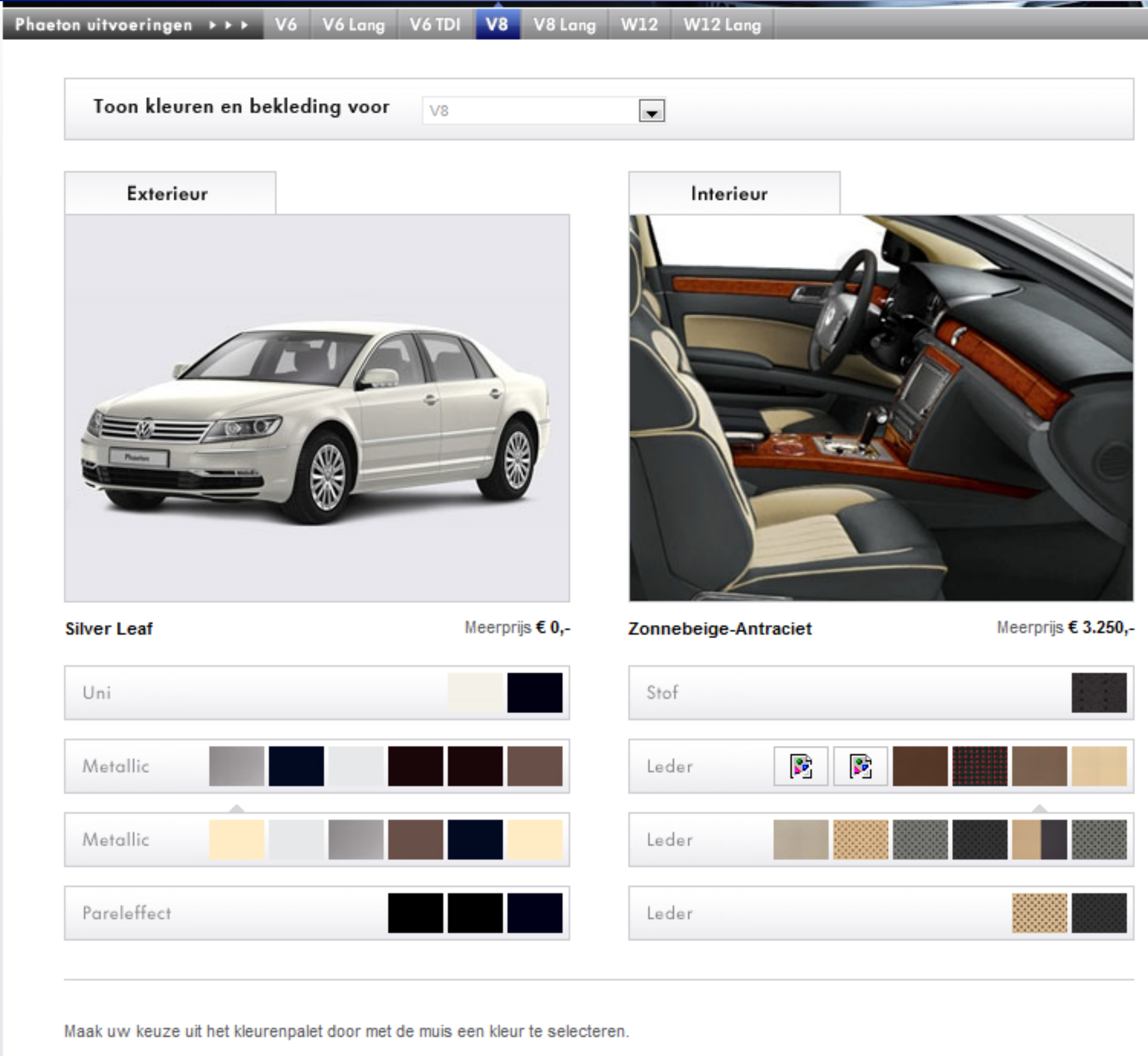
[Batory, IEEE TSE 2004]
[Czarnecki et al., 2000]

A product line of Lego minifigures (16 valid feature combinations)

‘Feature diagram’ for configuring the 16 valid products (allowing more…)



VW features


Phaeton uitvoeringen
V6
V6 Lang
V6 TDI
V8
V8 Lang
W12
W12 Lang
Toon kleuren en bekleding voor
V8
Exterieur
Interieur
Silver Leaf
Meerprijs € 0,-
Zonnebeige-Antraciet
Meerprijs € 3.250,-
Uni
Metallic
Metallic
Pareleffect
Stof
Leder
Leder
Leder
Maak uw keuze uit het kleurenpalet door met de muis een kleur te selecteren.

Configure your 11-inch MacBook Air

Hardware | Service and Support | Accessories | Printers

Hardware

Processor

Enjoy incredible performance from fourth-generation Intel Core processors. Choose the speed and processor you want.

Learn more

- (•) 1.3GHz Dual-Core Intel Core i5, Turbo Boost up to 2.6GHz
- ( ) 1.7GHz Dual-Core Intel Core i7, Turbo Boost up to 3.3GHz [+ £130.00]

Memory

More memory (RAM) increases overall performance and enables your computer to run more applications at the same time.

Learn more

- (•) 4GB 1600MHz LPDDR3 SDRAM
- ( ) 8GB 1600MHz LPDDR3 SDRAM [+ £80.00]

Storage

Your MacBook Air comes as standard with flash storage. Flash storage has no moving parts and provides faster responsiveness and enhanced durability.

Learn more

- (•) 256GB Flash Storage
- ( ) 512GB Flash Storage [+ £240.00]

Summary

£1,029.00 incl. VAT

Special 0% financing

Estimate Payments

Dispatched:
Within 24 hours
Free Delivery

Add to Basket

Gift package available

Contact Us

0800 048 0408

Live Chat

Specifications

1.3GHz Dual-Core Intel Core i5, Turbo Boost up to 2.6GHz

4GB 1600MHz LPDDR3 SDRAM

256GB Flash Storage

Backlit Keyboard (British) & User's Guide (English)

Configure your BMW vehicle

Are you interested in configuring your ideal BMW? Please select a country to visit the configurator in the Virtual Center or contact your local BMW dealer who will be happy to answer all your questions about the BMW model you are interested in.

United Kingdom

Related topics

‣ Request information

Order product catalogues, brochures and equipment lists direct from BMW.

‣ Arrange a test drive

Make a date for a test drive with your BMW dealer.

FIND YOUR BMW.

Filter

> Reset filter

+ Budget

− Vehicle type

- All
- Petrol
- Diesel
- Hybrid
- Electric Vehicle

− Body type

- Saloon
- Touring
- Convertible
- Coupé
- Gran Turismo
- Sports Hatch
- Roadster
- Sports Activity Coupé
- Sports Activity Vehicle

Number of seats

30 Vehicles (465 Model variants)

1

BMW 1 Series 3-door Sports Hatch (34)
from £ 17,775.00

BMW 1 Series 5-door Sports Hatch (39)
from £ 18,305.00

2

BMW 2 Series Coupé (14)
from £ 24,265.00

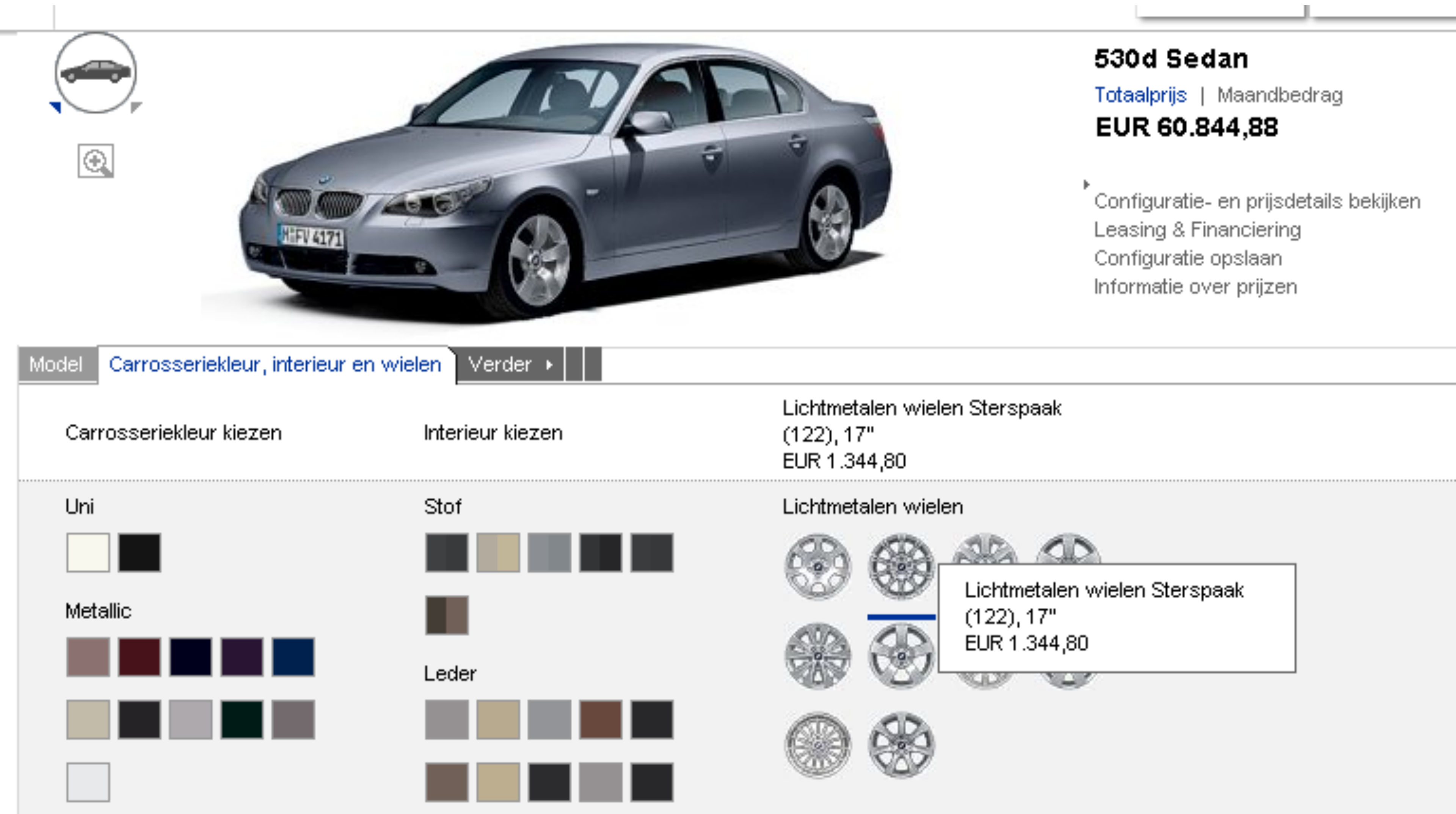
3

BMW 3 Series Saloon (56)
from £ 23,550.00

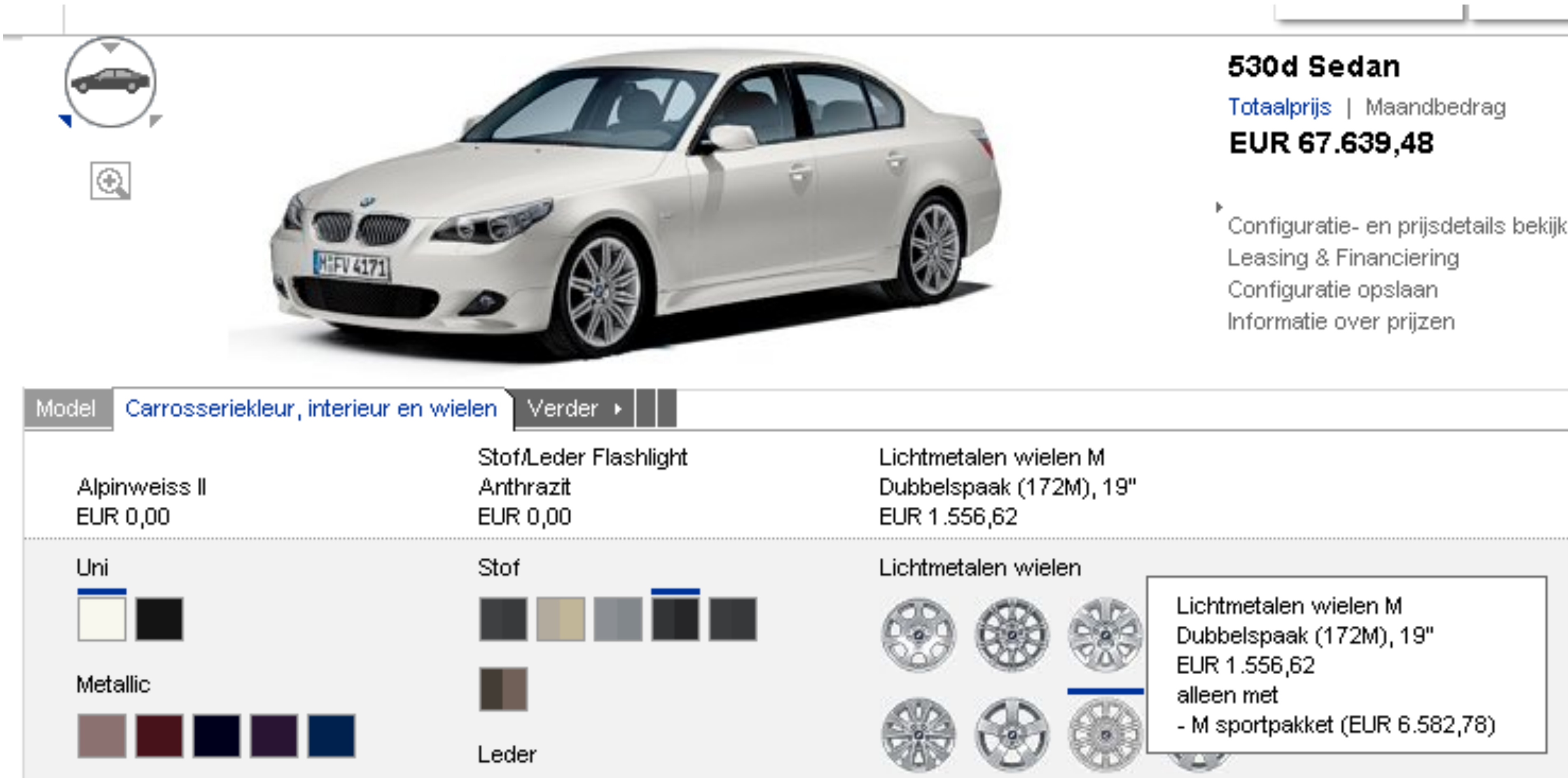
BMW 3 Series Touring (54)
from £ 24,865.00

BMW 3 Series Gran Turismo (39)
from £ 29,200.00

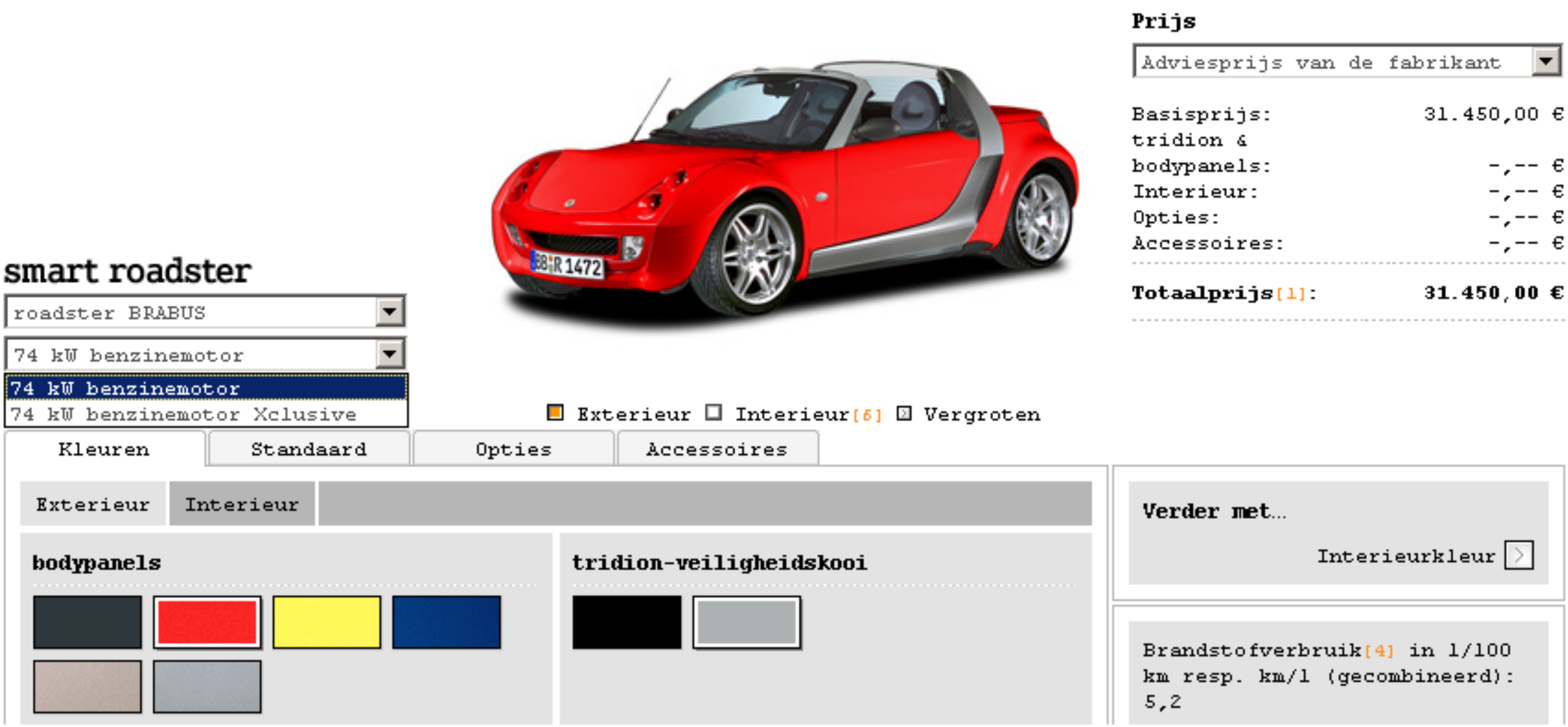
BMW product configurator



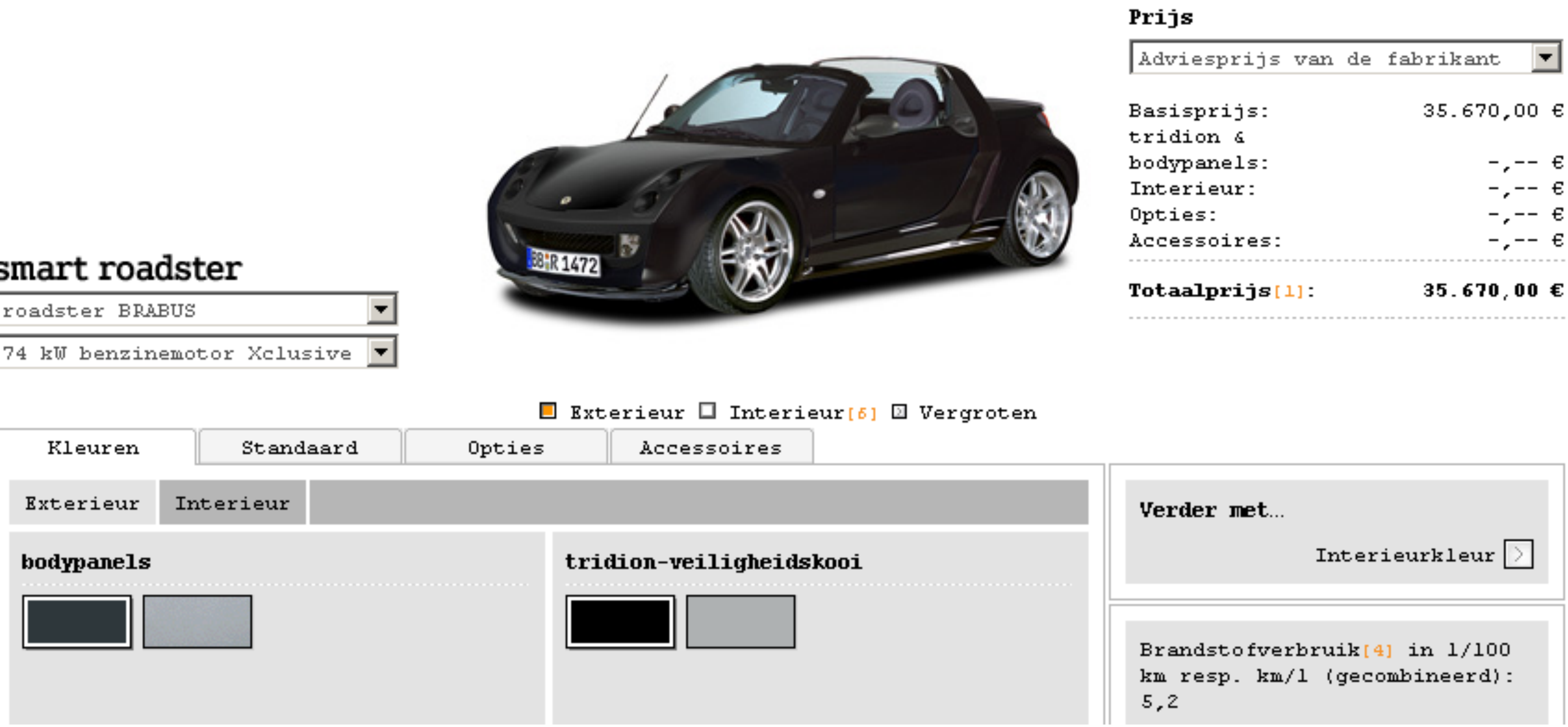
BMW feature packages



Smart constraints (1/2)



Smart constraints (2/2)



Feature binding times

- The moment a choice is made for a variable feature
- For a software product:
  - Compile time
  - Release time
  - Deployment time
  - Start-up time
  - Run-time
- For a car:
  - Spray time
  - Drive-out-of-the-factory time
  - …
- For lego:
  - Playing time 

Variation points and variants

- External variation points: visible to stakeholders mainly interested in observable properties like functionality and quality (e.g. customers or users)
- Internal variation points: visible mainly to stakeholders who develop the products (e.g. application designers or testers)
- Mandatory variation points *must* be selected for each product (i.e. some decision must be made)
- Optional variation points *may* be selected for a product (and a decision may be made)

Variability dependencies (1/2)

Allowed choices of variants at a specific variation point

- Mandatory variant: if the variation point is selected, this variant *must* always be selected
- Optional variant: if the variation point is selected, this variant *may* be selected but it does not have to be
- Alternative choice i.e. a collection of at least two optional variants together with a [min…max] notation to indicate the permissible number of variants to be selected: if the variation point is selected, at least “min” variants *must* be selected while at most “max” variants *may* be selected

Variability dependencies (2/2)

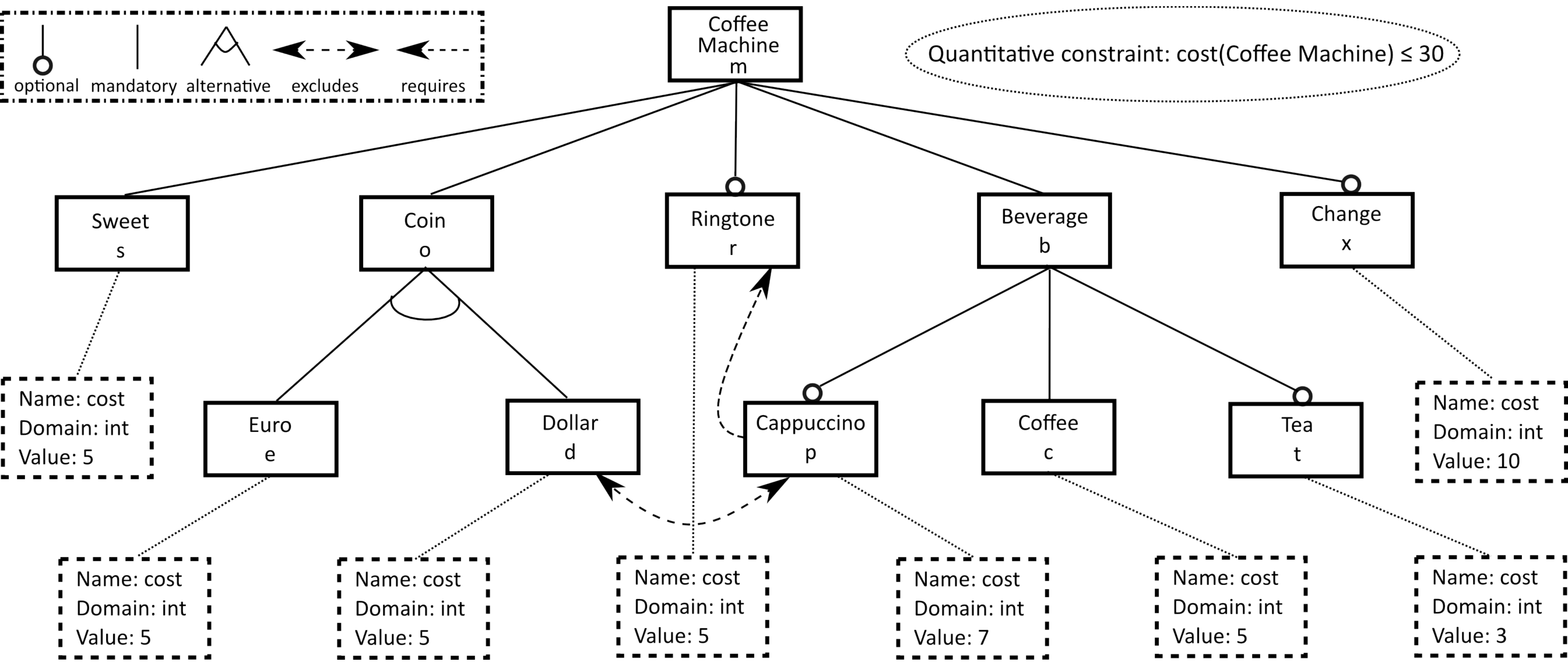
Cross-tree constraints

- Requires: indicates that the presence of one feature requires the presence of another feature
- Excludes: indicates that the presence of two features is mutually exclusive

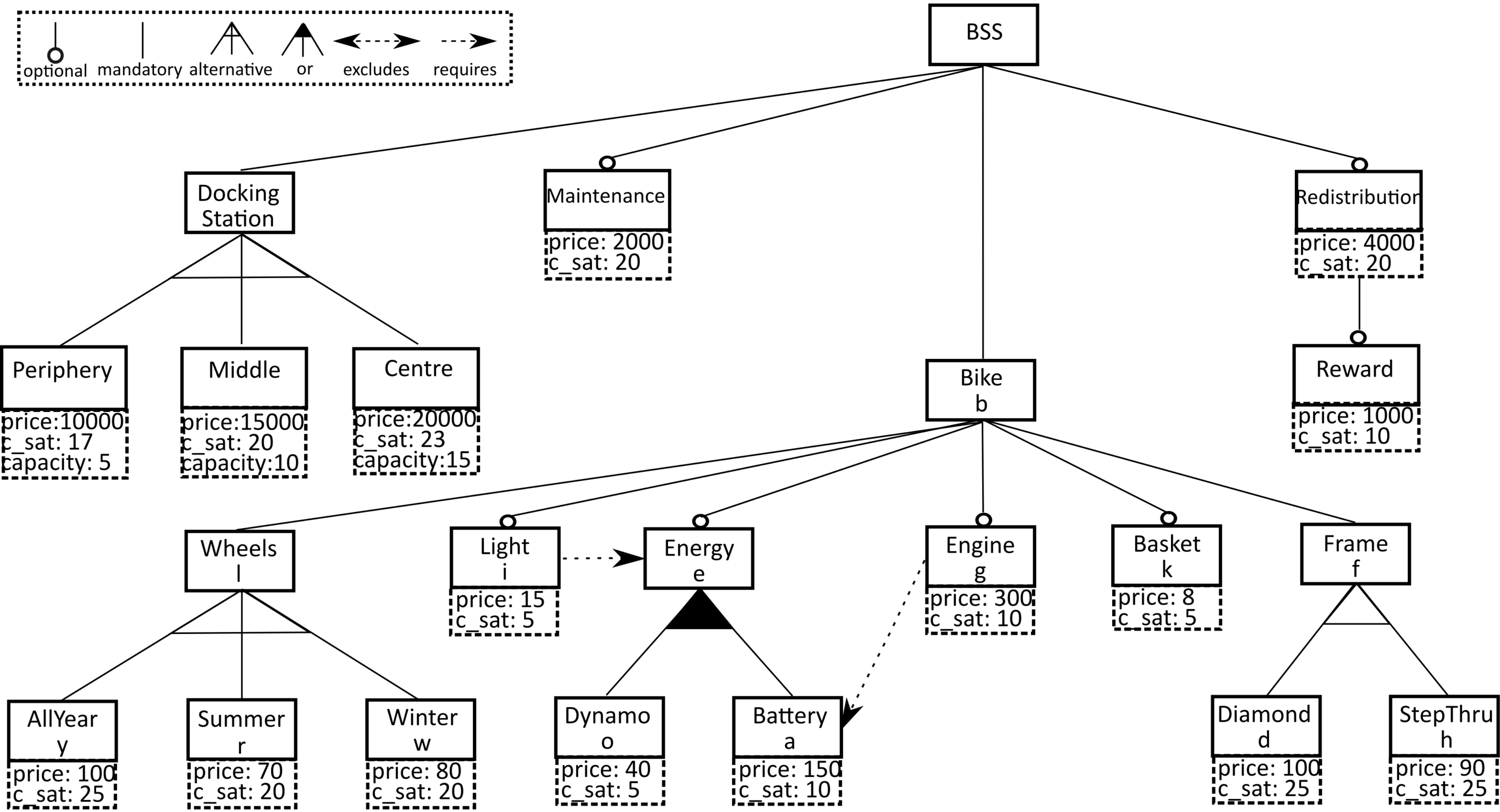
Quantitative constraints

- Feature attributes (non-functional): *cost*(feature) = 7, *etc.*
- *cost*(product) = ∑{*cost*(feature) | feature ∈ product }

Attributed feature model (1/2)



Attributed feature model (2/2)

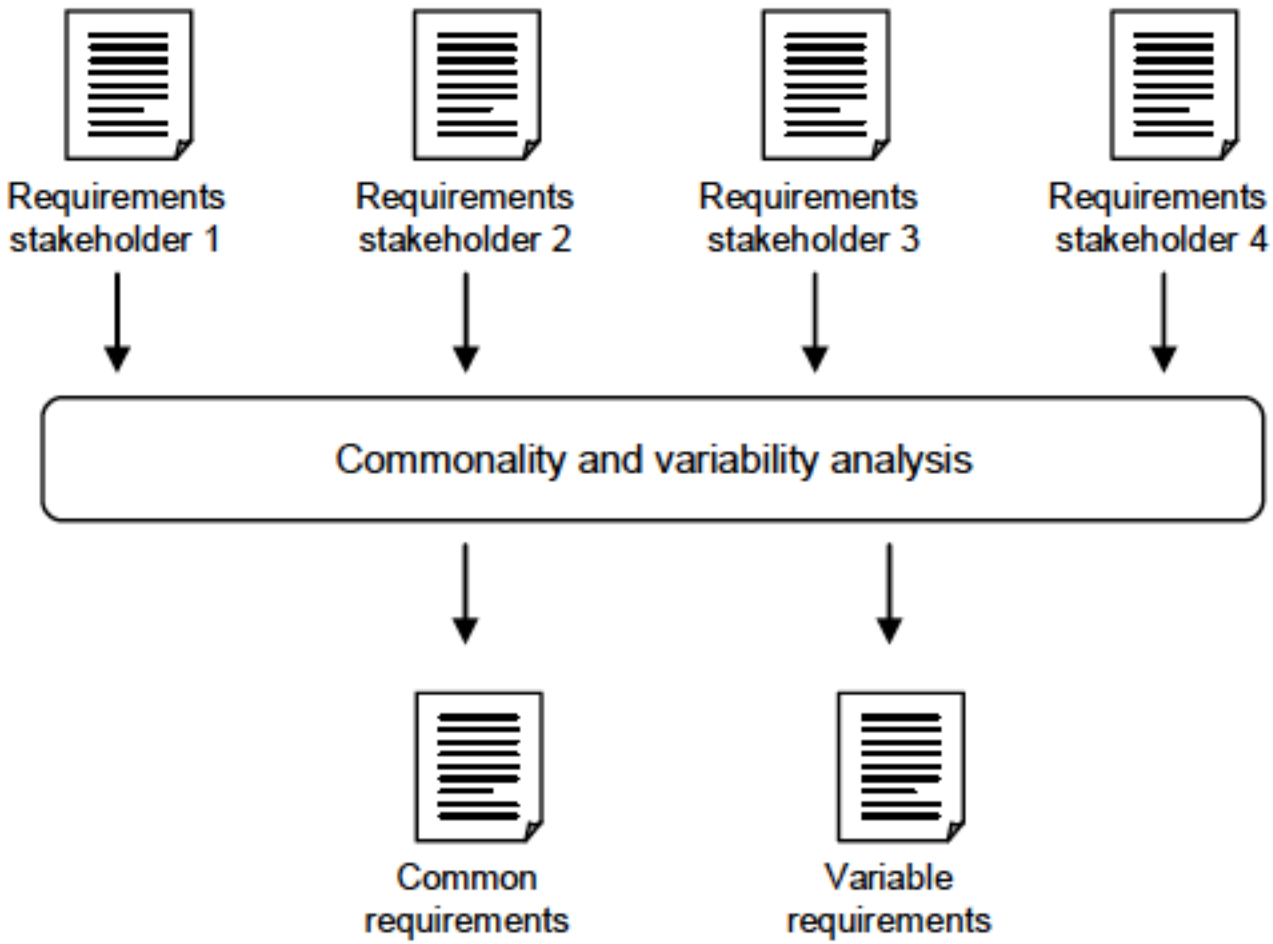


Domain RE

- During domain RE, the requirements for the entire SPL are defined
- Basis for: (i) developing the entire SPL family (ii) defining the requirements of each product
- Comprises the same core and cross-sectional RE activities as RE for single systems;

  these activities have the same goals plus, in addition, the goal to define the SPL variability

Elicitation of requirements variability



Application RE

- During application RE, the requirements for a specific application of the SPL are defined by exploiting the domain requirements artefacts (incl. defined variation points and variants)
- Compared with RE for single systems, two additional tasks must be accomplished:
  1. Binding the defined variability
  2. Documenting the variability binding

So why have variability?

- Flexibility to deliver
  - Different versions of products based on the same trunk of code (Windows XP professional, Windows XP Home edition, …)
  - Onto different platforms (a linux, a mac and a windows version)
  - With different components (Windows Media Player + Divx, iTunes, …)
  - Onto different plug-in products, *etc., etc., etc.*
- Is it an advantage then? Yes, but…
  - Requirements engineering becomes more complex
  - Deployment becomes more complex
  - Sales becomes more complex
  - Updates become WAY more complex
  - Testing becomes more complex (all configurations need to be tested)

"We always have 126,000,000 different bicycles in store!
(but only the parts for 1,000…)"

Weakness: feature interactions

- Features need to be combined, but have restrictions on each other
- Consider e.g. a phone switching system with the following features
  1. Call forwarding
  2. Do not disturb
  3. 3-way calling
  4. No interaction between the features
- Scenario 1: Bob forwards his calls to Alice; Alice sets "Do not disturb"; Bob receives a call, which is forwarded to Alice; Alice's phone rings!
- Scenario 2: Bob still forwards his calls to Alice; Bob invites Alice for a 3-way call; Carolina calls Bob to become part of the 3-way call; either
  1. Carolina is forwarded to Alice, who cannot accept any other calls then; or
  2. Carolina is accepted into the 3-way call

Weakness: complexity/scalability

with 33 optional, independent features

a unique product for every

person on this planet

320 optional, independent features

more possible products than estimated

atoms in the universe

(Behavioural) variability analysis

Rigorously establish critical system requirements (for quality assurance) with formal methods and automated analysis tools

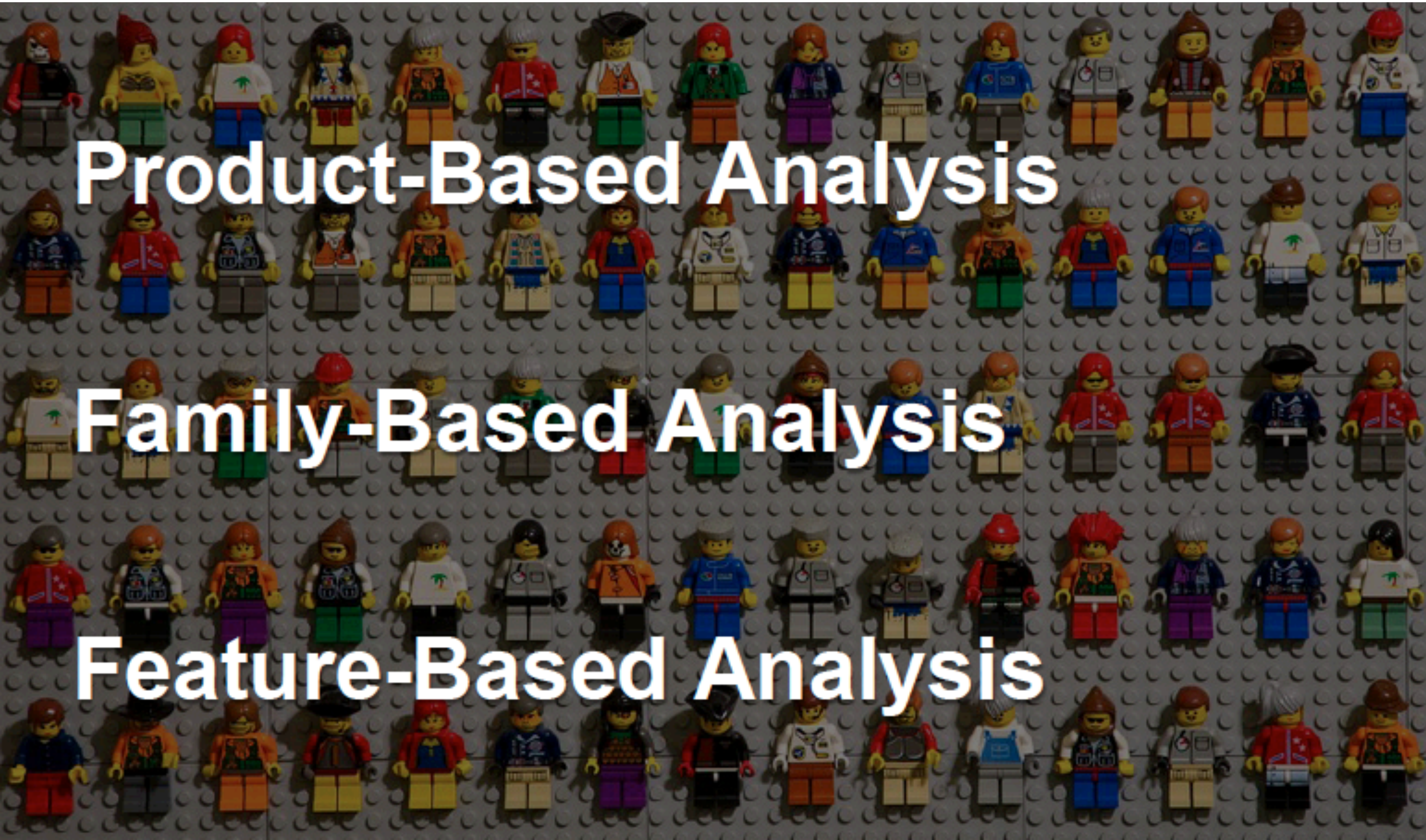
- For decades successful in single product/system engineering
- Not exploited broadly in SPLE, while correctness of artifacts for reuse and correctness of developed products is of crucial importance (many massively produced (embedded) systems and safety- or business-critical applications)

Traditionally

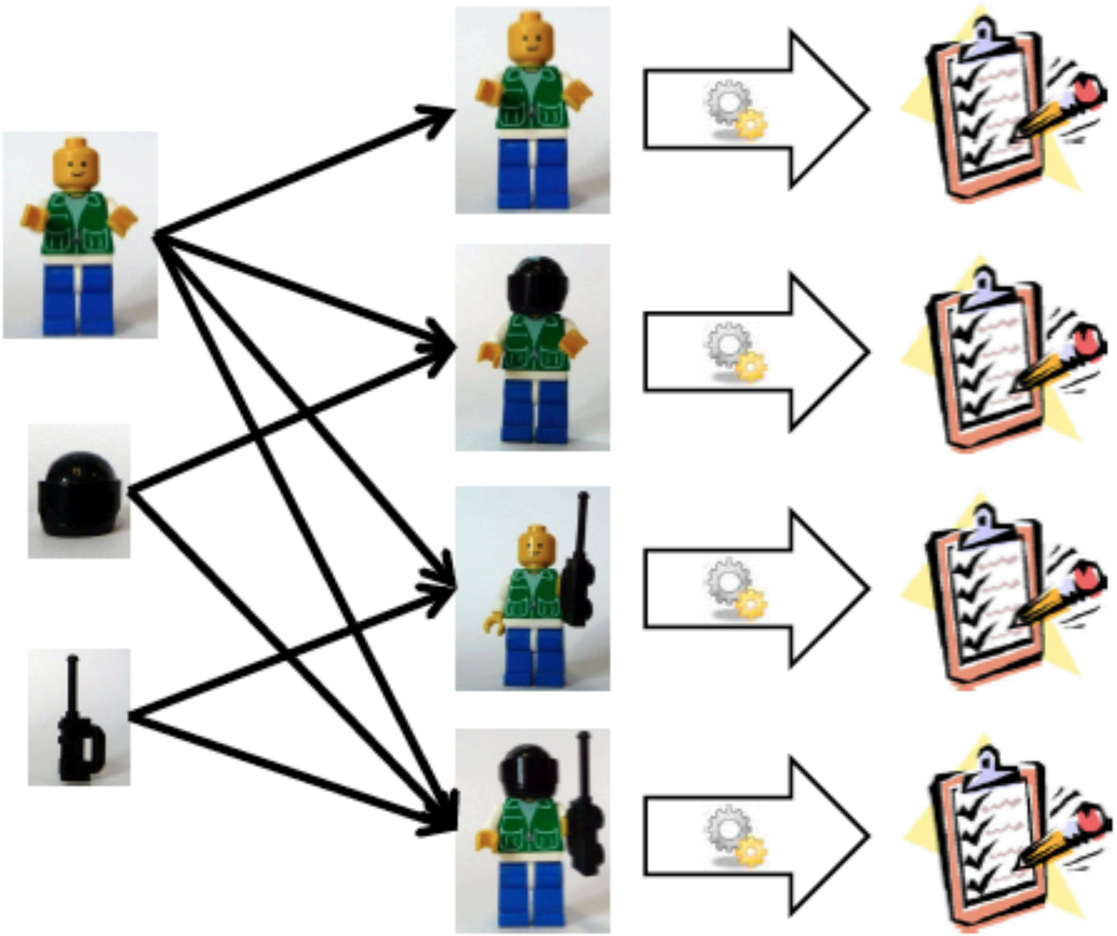
- Mainstream formal methods do *not* consider variability directly
- Formal methods that *have* been applied in SPLE mainly focus on structural rather than behavioural properties (feature model analysis, product line testing, *etc*.)

Variability analysis

(type checking, static analysis, model checking, theorem proving, testing, *etc*.)



Product-based analysis



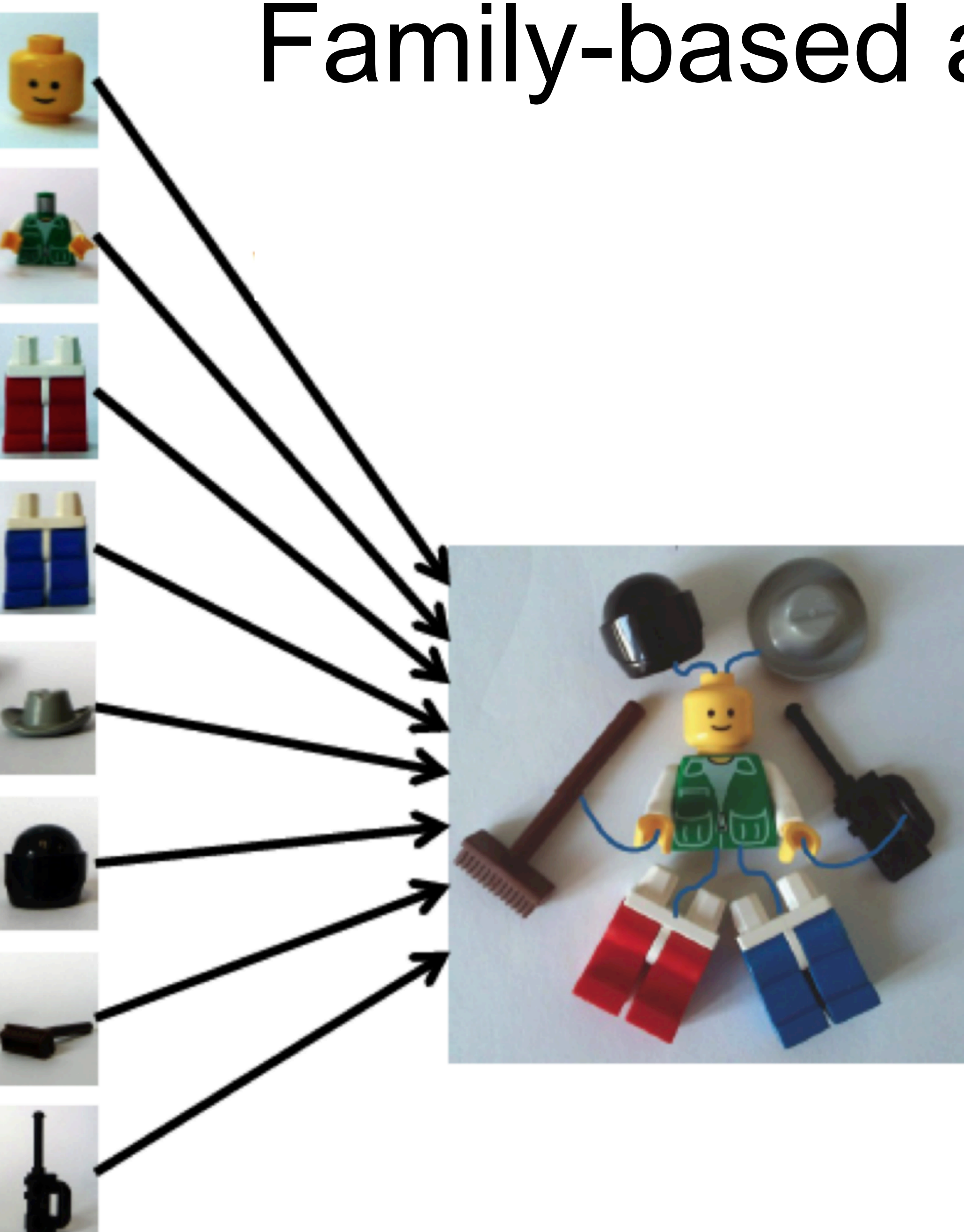
Simple approach

Standard tools available

Infeasible for large product sets

$O(2^n)$ for n features

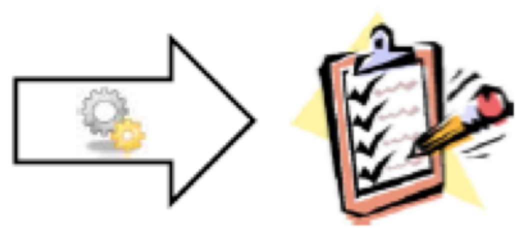
Family-based analysis



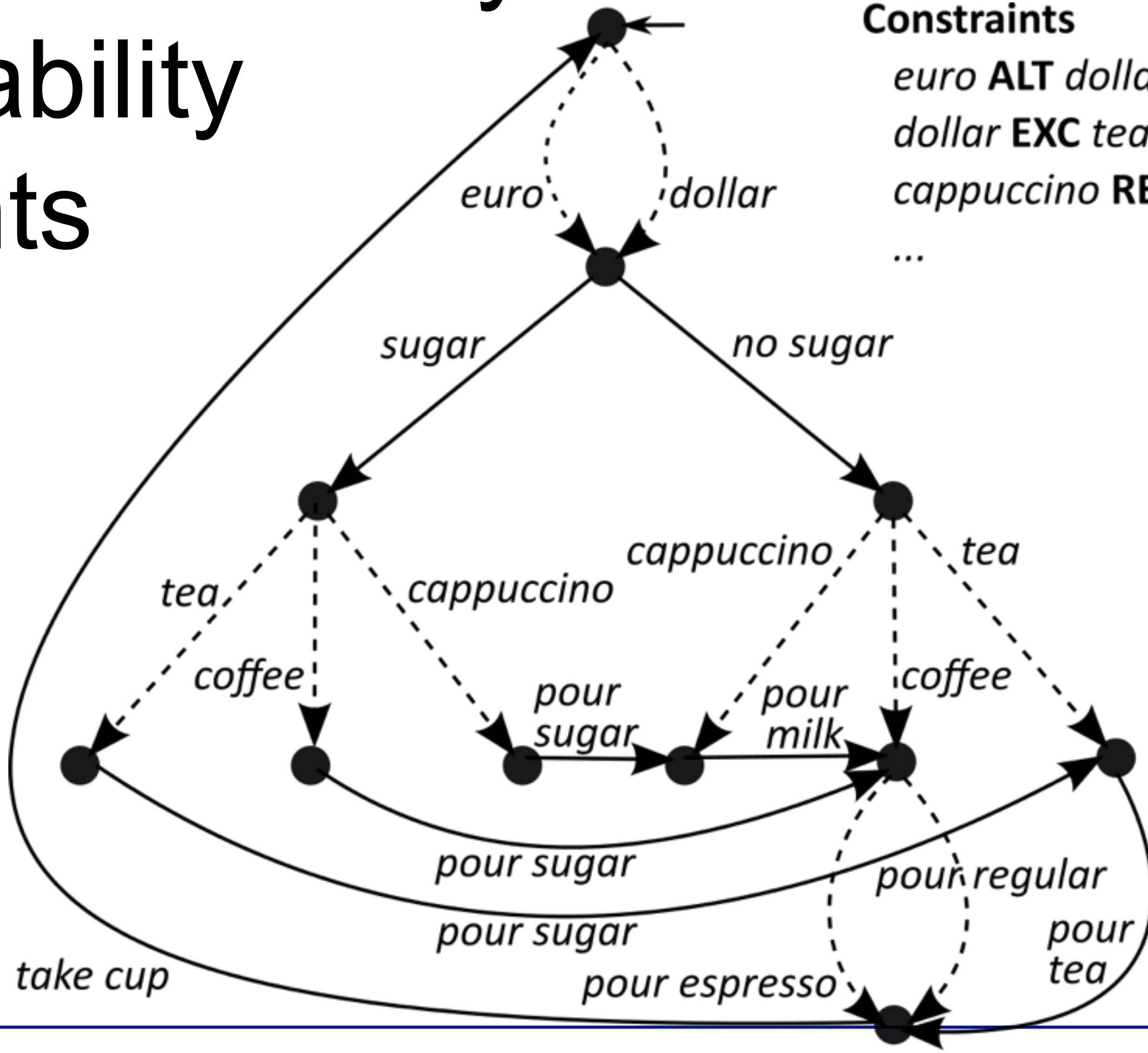
Beneficial for many products with substantial similarities

Generates complex analysis tasks

Requires (compact) family metamodels



Modal Transition Systems with variability constraints



Constraints

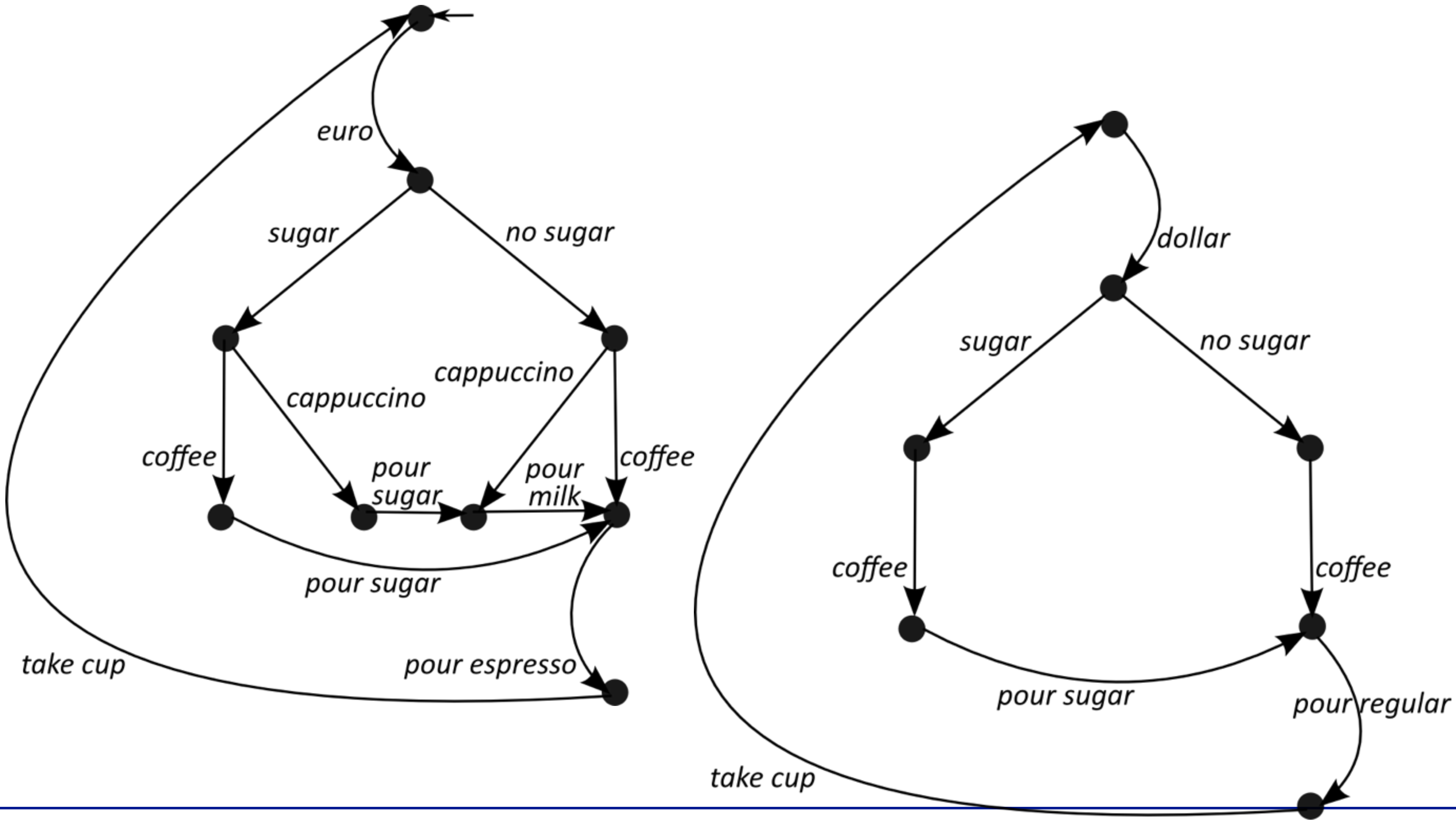
euro **ALT** *dollar*

dollar **EXC** *tea*

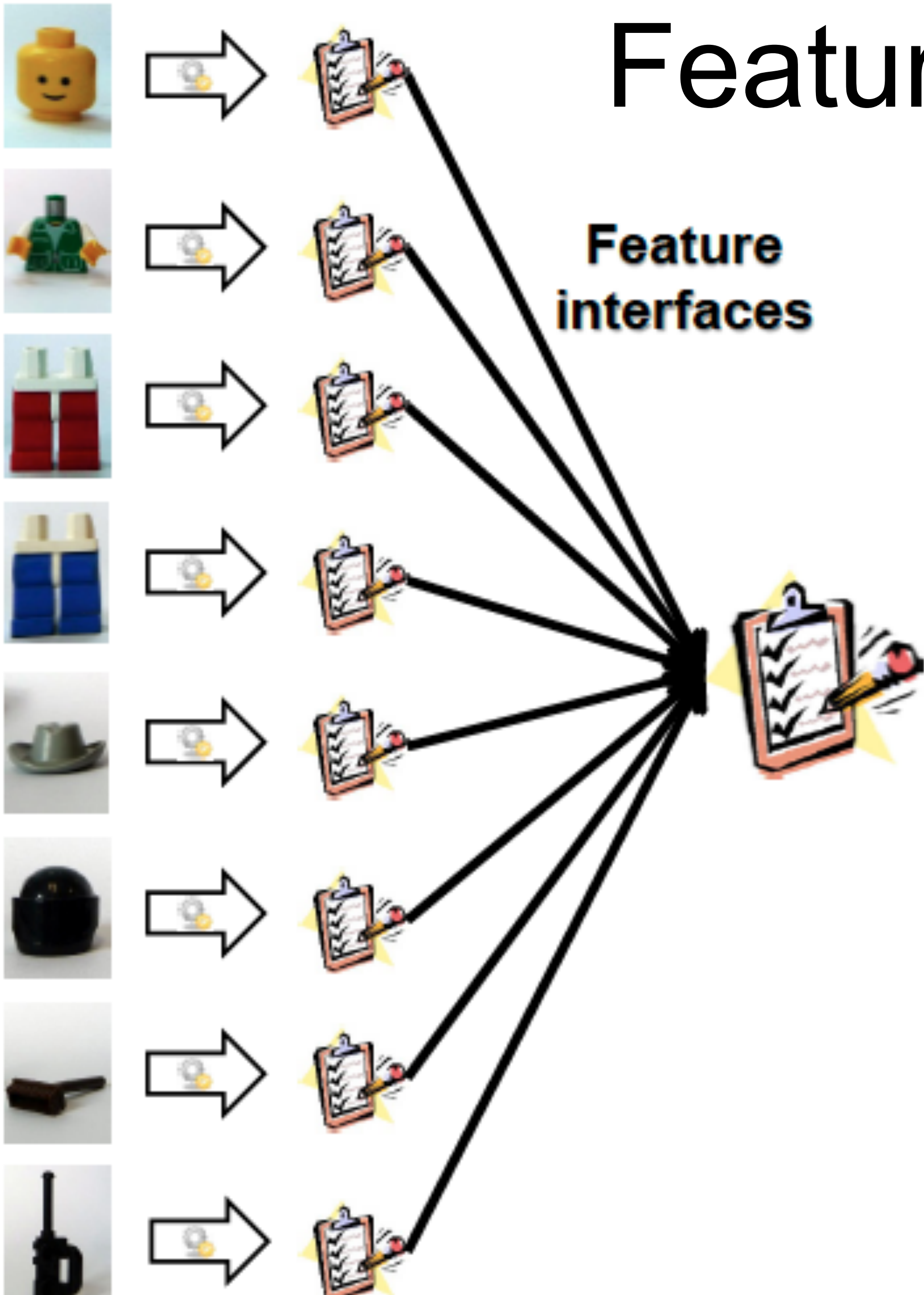
cappuccino **REQ** *coffee*

...

MTS ⊨ φ implies for all LTS ⊨ φ



Feature-based analysis



Beneficial for large and cohesive features

Support for open-world scenarios

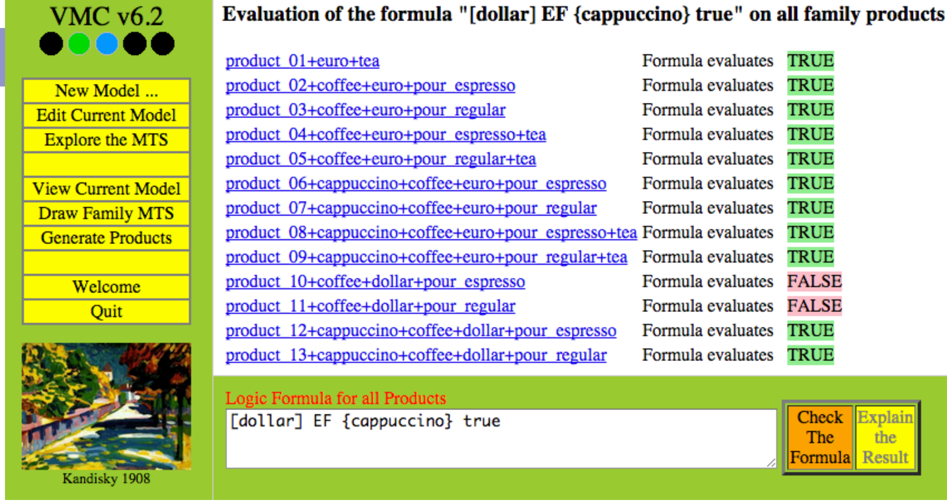
Incomplete w.r.t. feature interactions

Summary

- RE in SPLE differs from conventional RE
  - Separation of domain and application RE
  - Definition and binding of variability during RE
- Variability of an SPL is essential for its success

  (the available variability strongly influences which products can be developed based on the SPL according to customer wishes)
- Domain RE: elicit, document and validate variability
- Application RE: available variability communicated to the customer / user to bind the variation points and variants
- Central role of variability in SPLE: variability models/constraints
- (Behavioural) variability analysis: challenges existing techniques (growing numbers of products and increasing numbers of states)

Outlook



- An introduction to model checking
  - Temporal logics (LTL, (A)CTL, *etc.*)
  - Model-checking algorithms
- Compact models for product families
  - Modal Transition Systems with variability constraints
  - Featured Transition Systems
  - Model transformation: from FTS to MTS
- SPL model checking of MTS
  - v-ACTL: variability-aware ACTL
  - VMC: Variability Model Checker
- SPL model checking of FTS
  - A feature mu-calculus
  - Towards family-based model checking with mCRL2 language and toolset

mCRL2
analysing system behaviour