30. Idea Variation for a Mature Feature Model of the Product

Prof. Dr. Uwe Aßmann

Technische Universität Dresden
Software Engineering Group

Version 2020-0.3, 12.12.20
http://st.inf.tu-dresden.de

Business Network of the Day

- https://angel.co/

Obligatory Literature

- Kwanwoo Lee, Kyo C. Kang, and Jaejoon Lee. Concepts and guidelines of feature modeling for product line software engineering. Lecture Notes in Computer Science, 2319:62--78, 2002. Good overview on feature models
- Alexander Grots, Margarete Pratschke. Design Thinking — Kreativität als Methode. Marketing Review St. Gallen, April 2009, Volume 26, Issue 2, pp 18–23
    - DOI: 10.1007/s11621-009-0027-4
- Drew Boyd (Autor), Jacob Goldenberg. Inside the Box: The Creative Method That Works for Everyone. Profile Books Ltd. 2014.
    - Introduces Systematic Inventive Thinking (SIT)
    - http://www.sitsite.com/method/
    - https://en.wikipedia.org/wiki/Systematic_inventive_thinking

Any good business model (also an MVP)
should be improved by new variants or extensions.

Other Literature

- Don S. Batory. Feature models, grammars, and propositional formulas. In J. Henk Obbink and Klaus Pohl, editors, Software Product Lines, 9th International Conference, SPLC 2005, Rennes, France, September 26-29, 2005, Proceedings, volume 3714 of Lecture Notes in Computer Science, pages 7--20. Springer, 2005.
  - Explains the relationship of feature models and propositional logic.
- Hans de Bruin and Hans van Vliet. Quality-driven software architecture composition. Journal of Systems and Software, 66(3):269--284, 2003.
  - Introduces feature-solution graphs, the bipartite graph between feature trees and product-component trees.

Improving a BMC or Developing a New One?

- When a BMC has been graded and assessed, some its fields may need to be improved or ***varied (exchanged)***
  - Variation yields "greener" canvases

- A *red BMC* or a *red VPC* (failing the assessment) should be changed
  - If there is no successful sticky and viral MVV, there is a problem

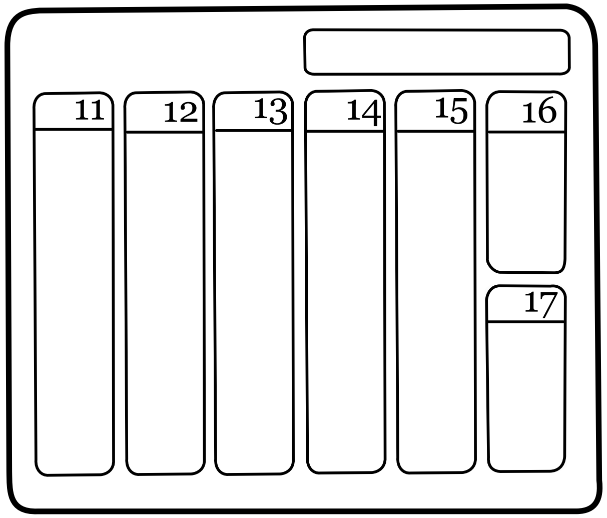
Point of Pivot: [Blank] Sometimes, this does not help and the BMC must be thrown away, and a **plan B** has to be found, another BMC.

- This chapter introduces systematic ways to change (vary, exchange) the fields of
  - Value proposition
  - Key resources
  - Customer segments

30.1. Canvases for Idea Generation

Shortcomings of Lean Startup from the Viewpoint of Software Product-Line Engineering

No support for consistent modeling of product lines
(no support for feature modeling and feature variation)



No support for canvas modeling (composition and engineering)

No support for staged feature configuration with suppliers

No support for grading and metrics

Remember the Value of the Variation-Based Business Model (Software Product Lines)

„**Software product lines** represent perhaps the most exciting paradigm shift in software development since the advent of the high-level programming languages. Nowhere else in software engineering have we seen such breathtaking improvements in cost, quality, time to market, and developer productivity, often registering in the order-of-magnitude range.“

„At the Software Engineering institute, we have recorded case study after case study of companies succeeding in one market area with a product line approach, and then taking their production capability to a nearby, under-exploited area of the market, and quickly rising to market dominance in that area as well. And why not? **If you can outperform your competitors by order-of-magnitude levels**, it's hard to imagine what could you keep from becoming a market leader.“

Paul Clements, SEI, in „Software Product Lines in Action“, Springer-Verlag.

Techniques for Idea Generation and Their Canvases

- A canvas can be used for scaling a business

BMC Variation

- For systematic variation of the fields of BMC
- with 4 structured operations of the book „Business Model Generation"

S.I.T. Canvas (Inside-the-box canvas)

- For systematic variation with Systematic Inventive Thinking (S.I.T.)

SCAMMPERR Canvas

- Structured process for idea variation

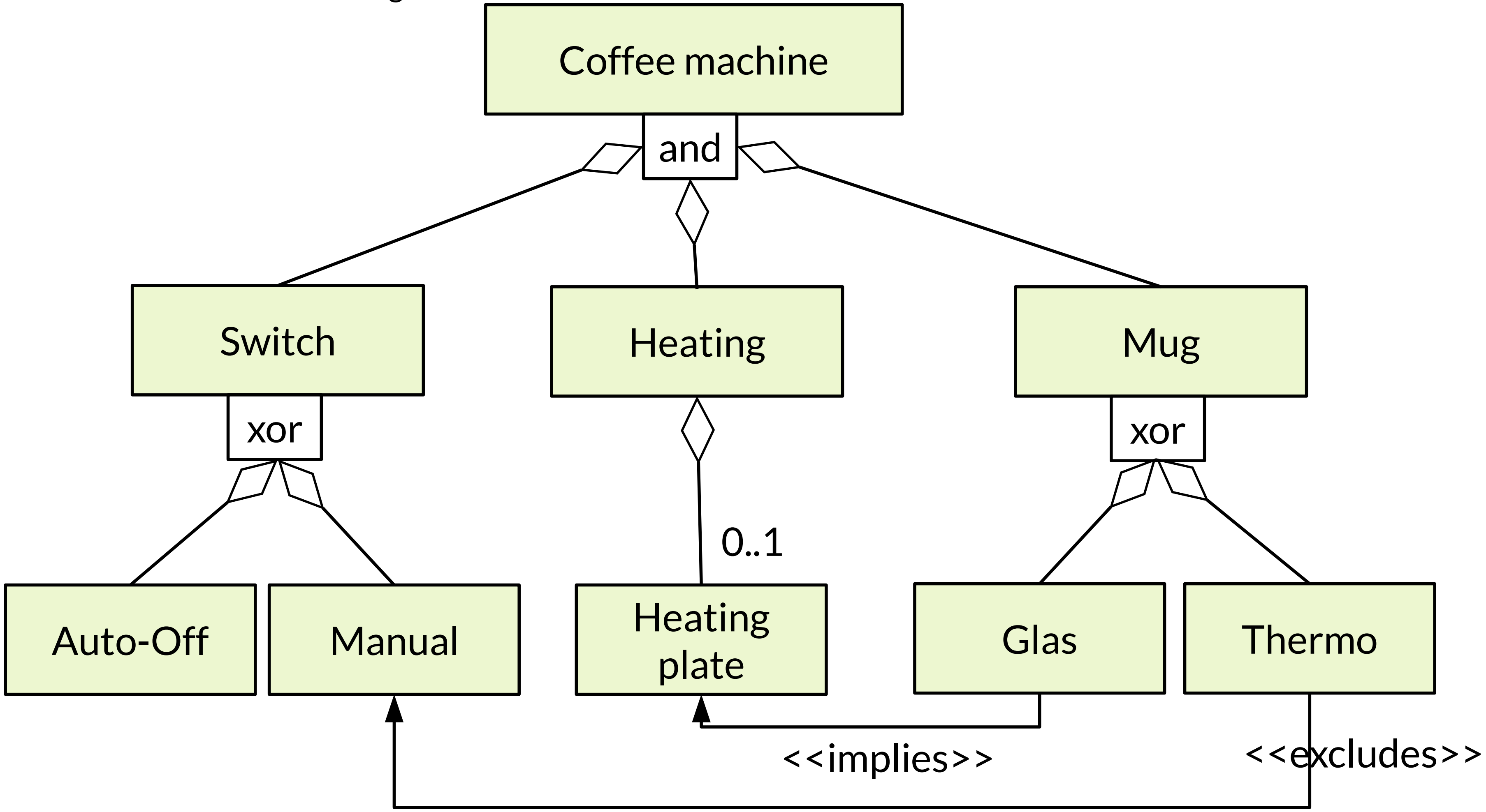
Hexagon Variation Canvas

- Structured process for variations
- Priorization of variants
- For scaling

30.1.1. Product Component Trees and Feature Trees

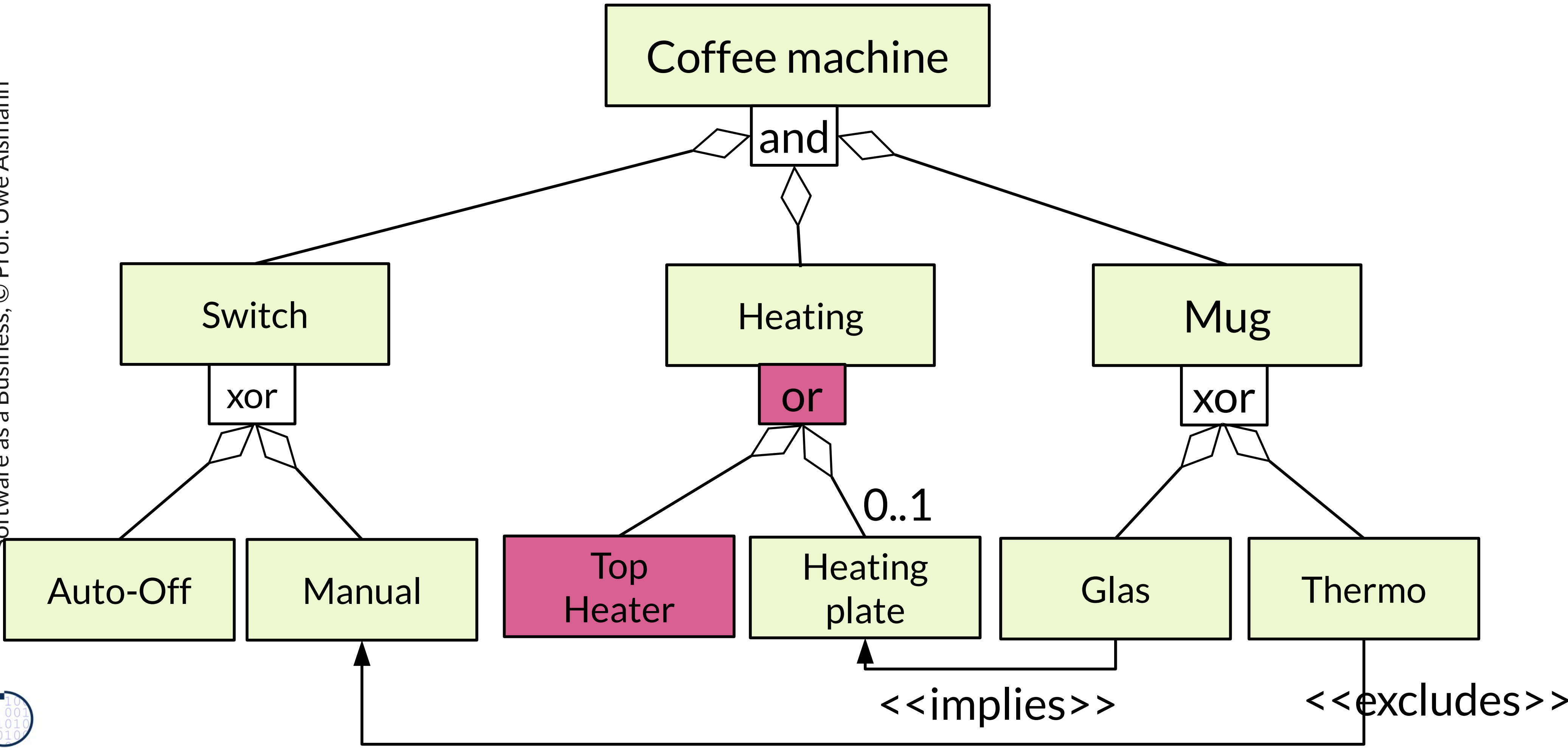
Product Component Tree

- A **component tree** is a and/or link tree of the components of a product, with options, inclusion and exclusion constraints.
  - It describes a combinatorial variant space of *components* and can be mapped to propositional logic
  - Product Component Trees generalize Product Breakdown Structure (PBS) from Course Softwaremanagement



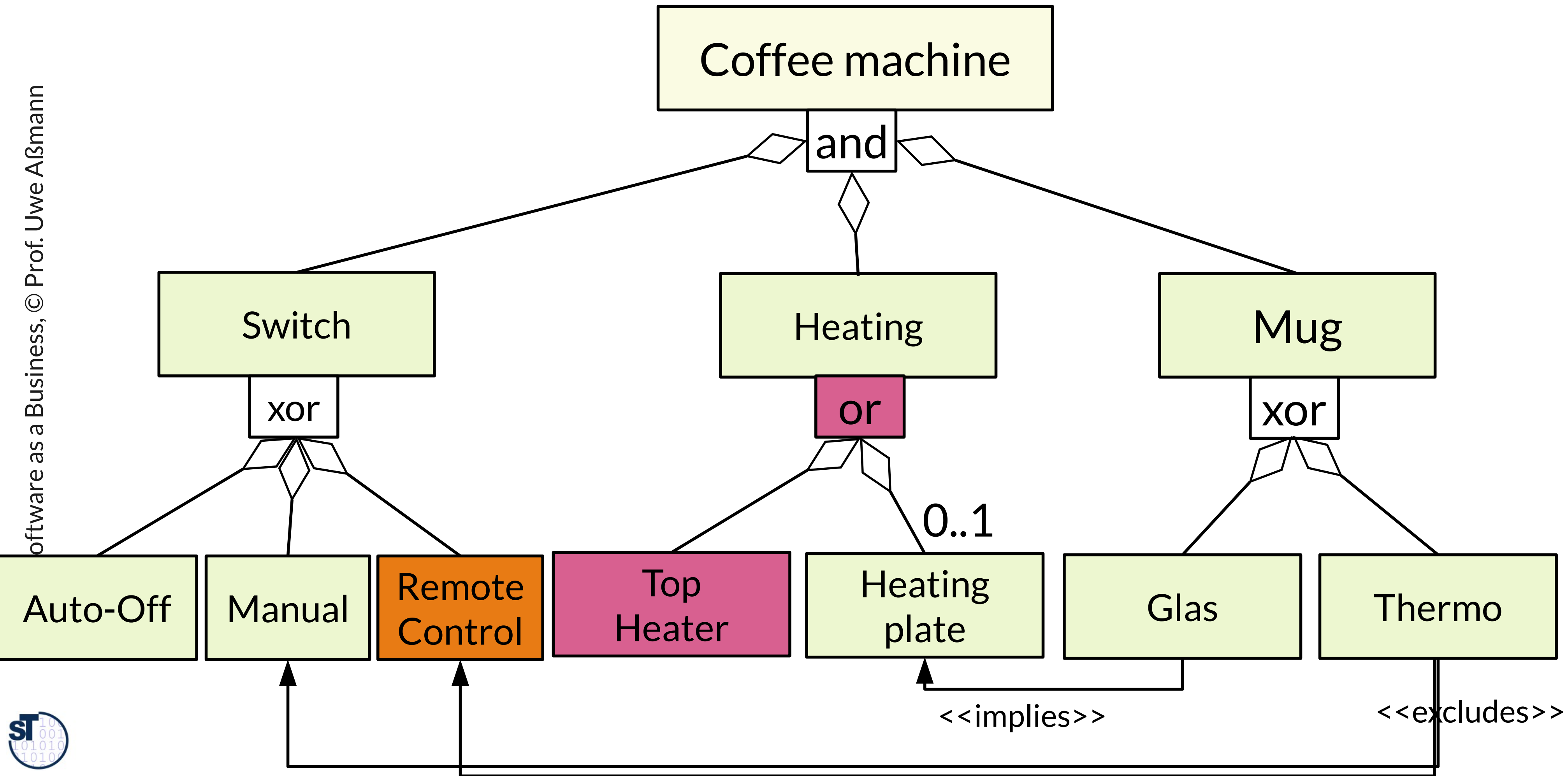
Varianting Means to Add Alternatives to a Product Component Tree

- Variation adds
  - new alternatives to an OR or XOR node
  - New OR or XOR nodes to AND nodes



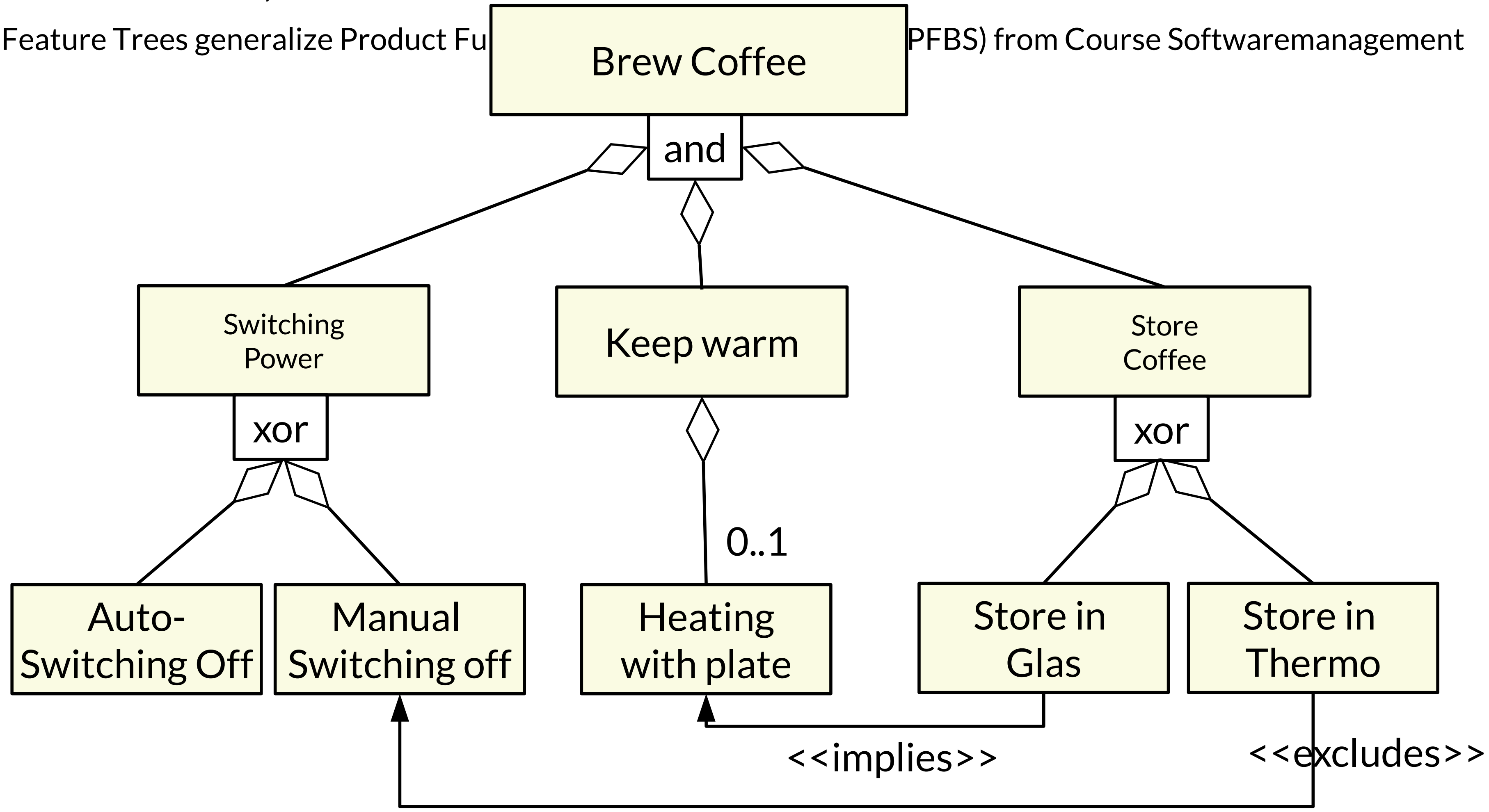
Varianting Means to Add Alternatives to a Product Component Tree (2)

- Step by step, new components (for new features) can be added



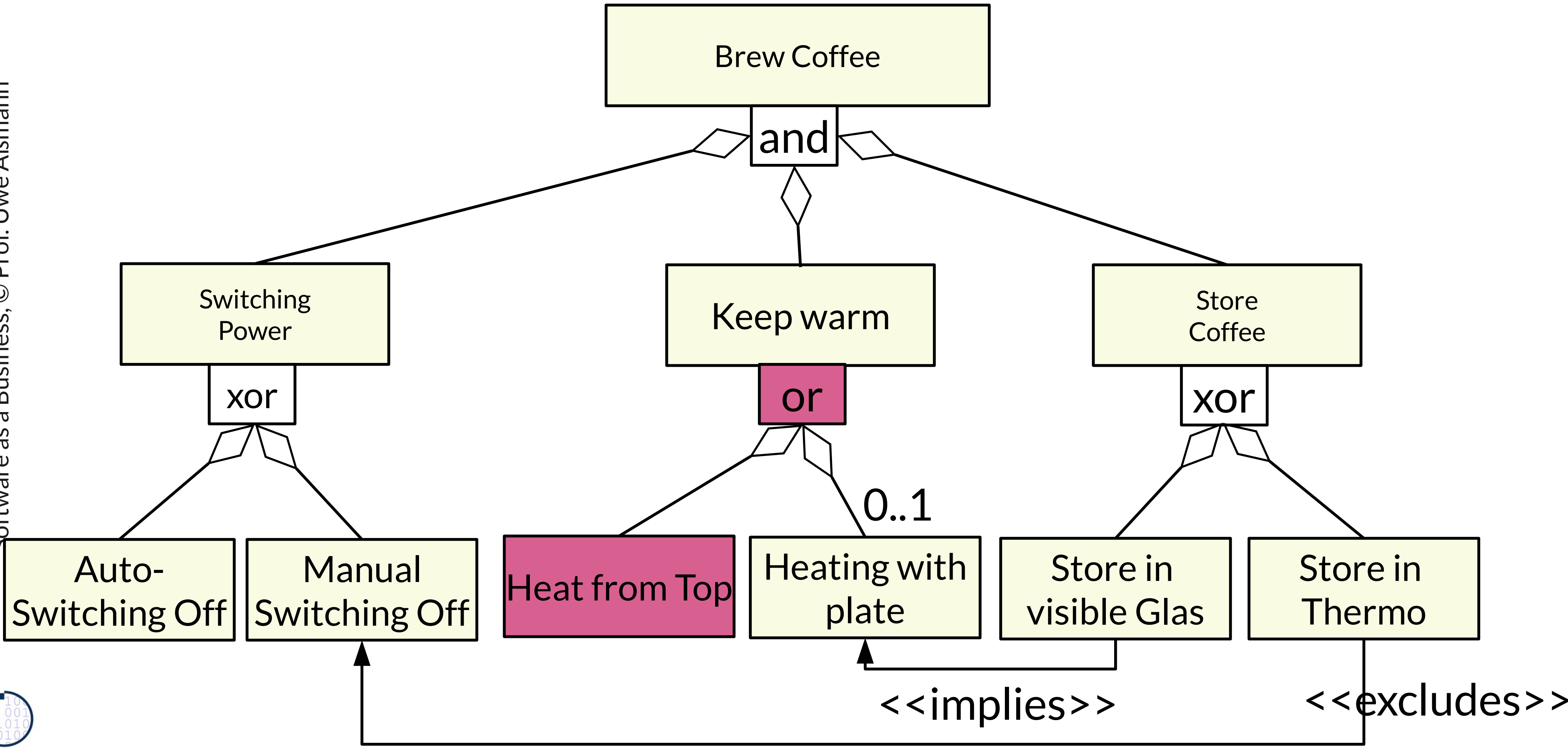
Feature Tree (Feature Model)

- A **feature tree (feature model)** is a and/or link tree of *functions (features)* with options, inclusion and exclusion constraints.
  - Functional decomposition
  - It describes a combinatorial variant space of *functions* and can be mapped to propositional logic over *functions*
- Feature Trees generalize Product Fu PFBS) from Course Softwaremanagement



Varianting Means to Add Alternatives to a Feature Model

- Variation adds new **feature alternatives** to an OR or XOR node
    - New OR or XOR **feature nodes** to AND nodes
- Attention: feature trees are not component trees!



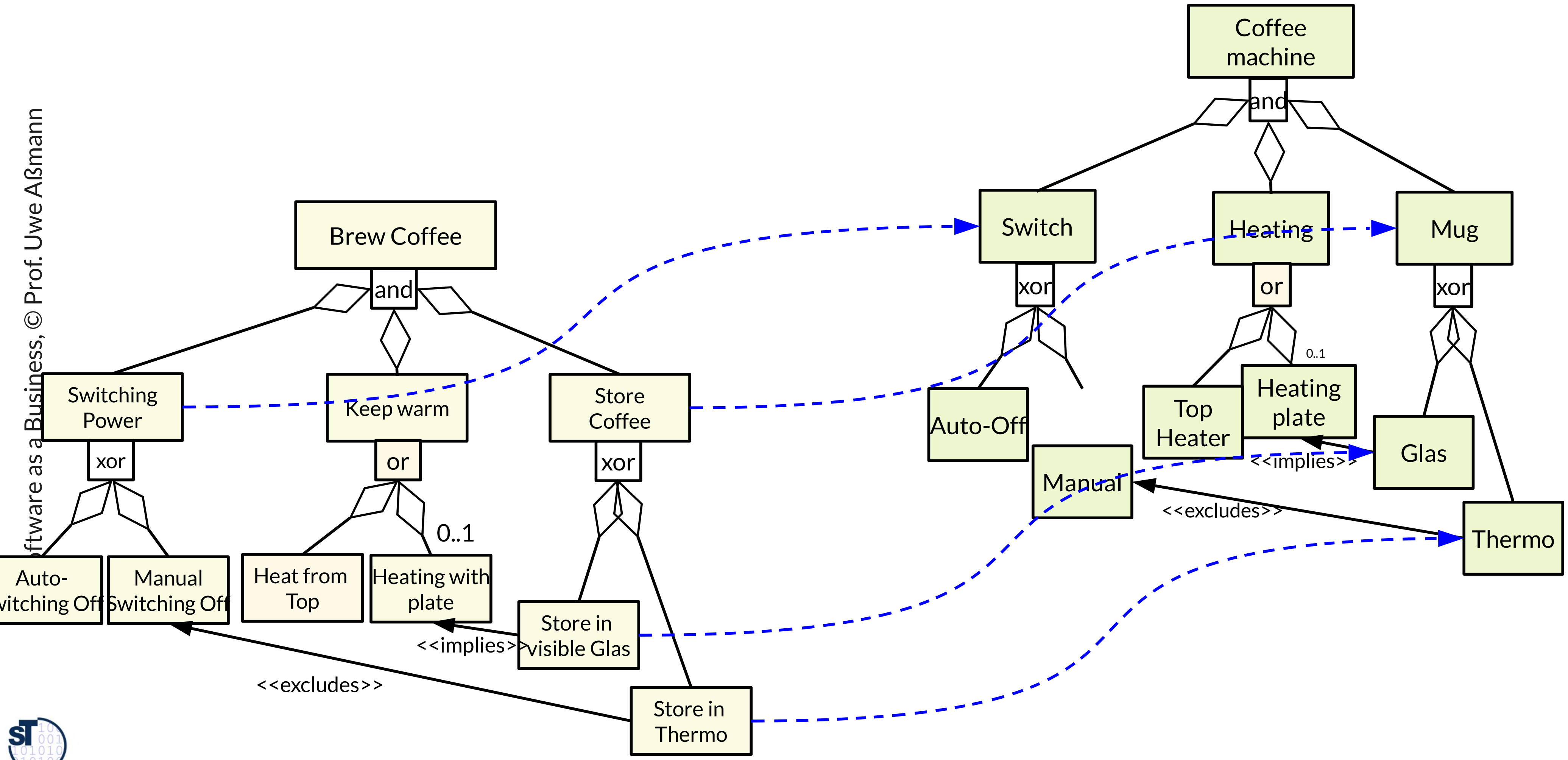
Idea and Feature Variation with Feature Trees

- Business model:
    - Product-oriented
    - Product-line-oriented
    - Software ecosystem: Features are distributed into apps on a software platform
- *Exercise: How to extend the features of a coffee machine?*

Feature-Solution Bigraphs

A ***feature-solution (bi-)graph*** maps a feature tree to a product tree via a *feature-solution mapping*

- Bigraphs contain two or more graphs (dimensions) linked by a *link graph*



Warning - The Scale Trap

- Many companies start *without* feature tree, product component tree, and feature-solution graph
  - -> They have a hard time finding and developing the components of the product, as well as their integration
- After 2 years, when they want to scale, they change to a product-line business model
- Feature trees and product component trees are *indispensable* for the management of a product line
- Then, the feature tree has to be *reconstructed*

Law of scaling:
If you want to scale:
Maintain a feature tree and a product component tree, as well as their feature-solution mapping
from the MVV on

Next Pitches

On Fri, Dec 18, 2020, there will be the „landing page and smoke video“ pitch.

Please also prepare a component tree or feature tree to motivate how you can scale.

On Fri, January 22, 2020, there will be the first MVP pitch.

Please also prepare a component tree or feature tree to motivate how you can scale.

On Mon, Febuary 1, 2021, 16:40, there will be the „Dungeon of Dragons“.

30.2. Change-Driven Invention of new Products Variability-Based Design

- For scaling, it is important to develop *alternatives* and *variants*

- Drew Boyd, Jacob Goldenberg. Inside the Box. Why the best business innovations are right in front of you. Profile Books, London, 2013
- http://en.wikipedia.org/wiki/Systematic_inventive_thinking
- http://en.wikipedia.org/wiki/Unified_structured_inventive_thinking

30.2.1 Business Model Development with 4-field Portfolio of BMG, as a Matrix Analysis

[BMG p 231]

Matrix Analysis with 7W

- A Matrix Analysis combines two dimensions, for one canvas a set of questions or concepts
- With a matrix analysis, we create new ideas

Questions 7W	Key Partners	Key Activities	Key Resources	Costs	Value Propositions	Customer relationships	Channels	Customer Segments	Revenues
Who?									
What?									
When?									
Where?									
Why?									
What for?									
How?									

4-Actions Variation Framework of BMG

- The BMG book presents 4 operators for new ideas:
  - Eliminate
  - Reduce
  - Augment
  - Create
- To model the influcence of these dimensions of the BMC, we have to span up a matrix with 4x9 elements (matrix analysis)

Matrix Analysis for 4-Actions-BMC

- For this aspect-oriented matrix analysis for the BMC, create a table (matrix) of 4-actions and BMC, brainstorm on the crossproduct

	Key Partners	Key Activities	Key Resources	Costs	Value Propositions	Customer relationships	Channels	Customer Segments	Revenues
Eliminate									
Reduce									
Augment									
Create									

[BMG p.233ff]

Matrix Analysis for BeNiSiLo-BMC

- For this aspect-oriented canvas analysis on AUGMENT, create a table (matrix), brainstorm on the crossproduct
- The "operations dimension" is BeNiSiLo, a quality-oriented set of improvement operations

Augment	Key Partners	Key Activities	Key Resources	Costs	Value Propositions	Customer relationships	Channels	Customer Segments	Revenues
Better									
Nicer									
Simpler									
Longer Lasting									

30.2.3. Variability-Based Business with Systematic Inventive Thinking (SIT)

SIT operates on Component Trees (Component Tree Algebra)

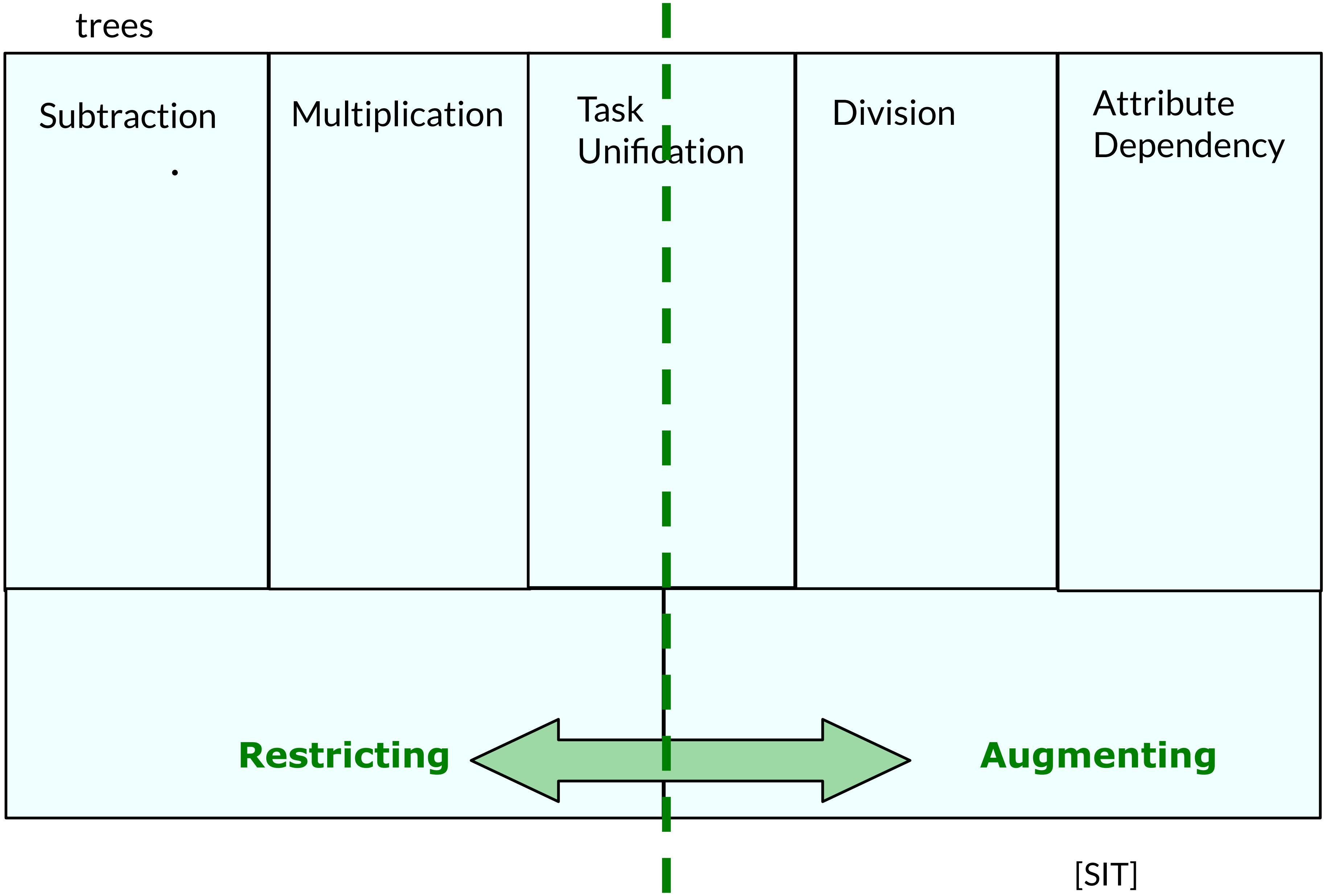
[https://en.wikipedia.org/wiki/Systematic_inventive_thinking]

- SMUDAD-operations on products and their component trees
- Also possible on feature trees

S	Subtract (Eliminate)	Remove, subtract components, reduce to core („Steve Jobs pattern“)
M	Multiply	Add another component, potentially different component to the product
U	Unify tasks	Find a new task for a component so that it can deliver two tasks
D	Divide	Re-group the components of the product into subgroups and form a new product (product out-lining). A first step to a product-line oriented business model
AD	Attribute dependency	Remove or create dependencies between parameters of components

SIT Canvas with SMUDAD Operations

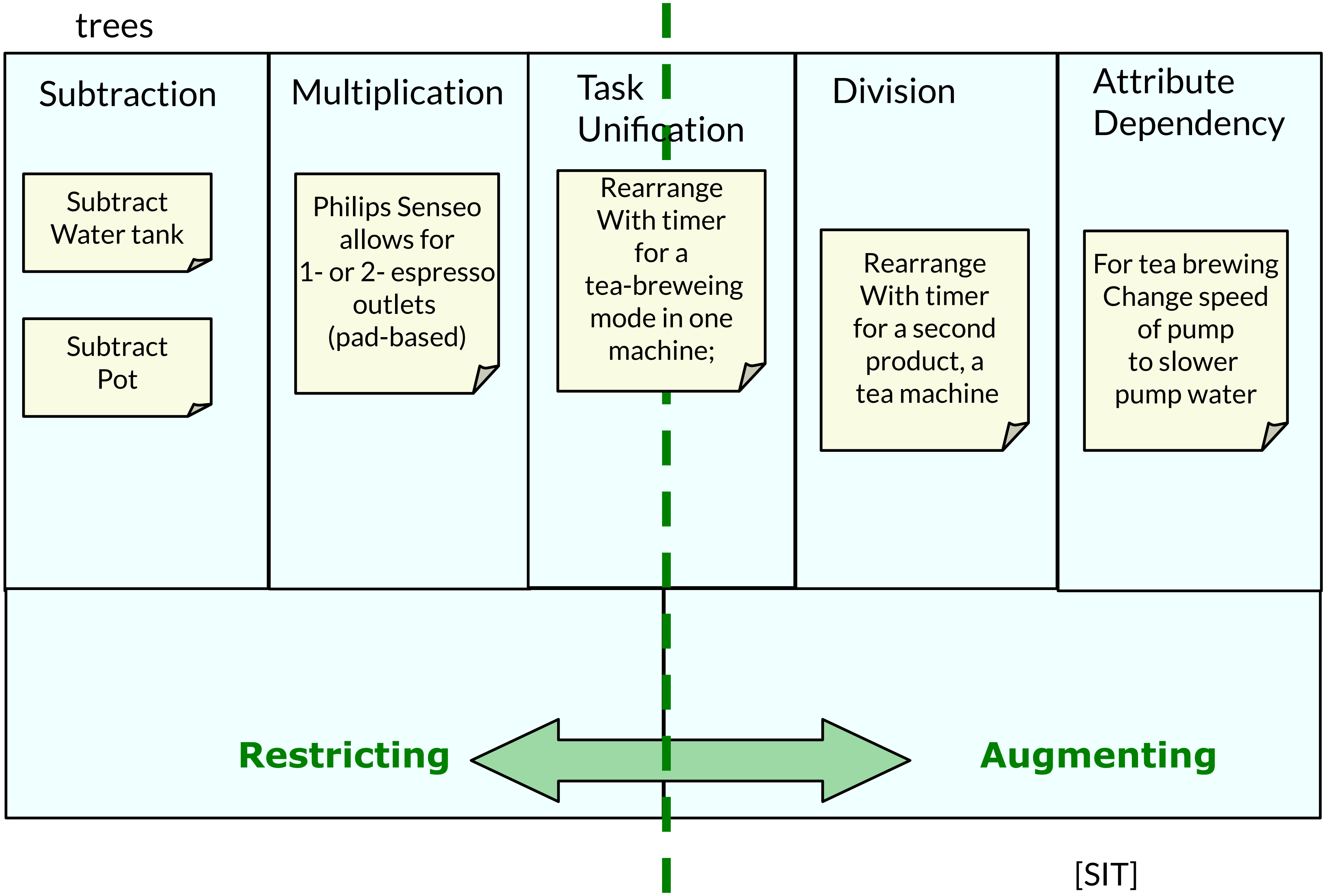
- SIT Canvas is based on simple modification operations of existing product component trees



[SIT]

Example: SIT Canvas on Coffee Machine

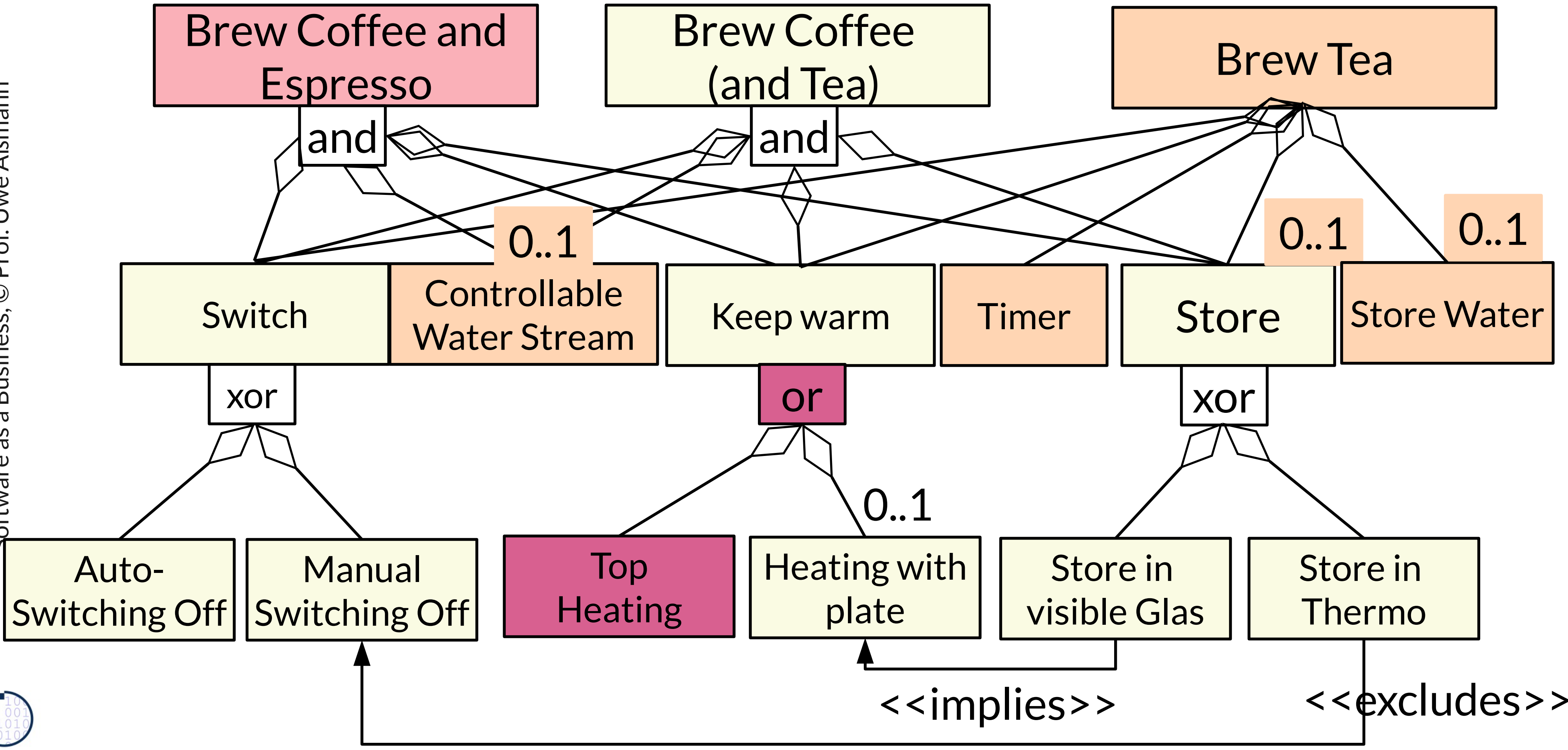
- SIT Canvas is based on simple modification operations of existing product component trees


Subtraction
Subtract Water tank
Subtract Pot
Multiplication
Philips Senseo allows for 1- or 2- espresso outlets (pad-based)
Task Unification
Rearrange With timer for a tea-breweing mode in one machine;
Division
Rearrange With timer for a second product, a tea machine
Attribute Dependency
For tea brewing Change speed of pump to slower pump water
Restricting
Augmenting

[SIT]

Extended Feature Model (Multihierarchical Feature Model of Product Line)

- Variation adds 2 new products (Tea machine, coffee+pad-espresso machine)
- CoffeeMachine with enriched feature set
- Feature model may become too complex → refactoring necessary



SIT thinking **Subtraction** Technique

- Subtracting components from the component set of a product
- Implied: removing features from the feature set of a product
  - Make it simpler and easier to use
  - Reduce costs

Examples:

- Steve Jobs was great in subtractions
  - Ipod with very few knobs
  - Ipad has no keyboard (compare to Microsoft Surface)
    - No USB
    - No CD/DVD

Division (Decomposition) Technique

- A CD, radio, cassette player all contain amplifyers.
- An *integrated music center* contains a CD, radio, cassette player and amplifyer.
    - One amplifier provides amplification for every other device.
    - Function is *divided*
- A *modular music center* is composed of components that can be replaced
    - Function is divided and replacable

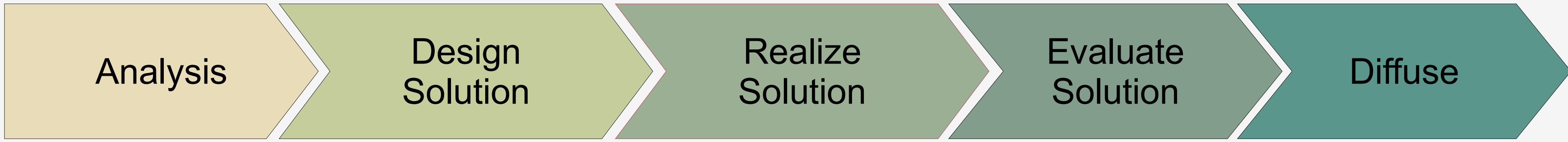
Matrix Analysis SITxBMC

- For this aspect-oriented canvas analysis, create a table (matrix), brainstorm on the crossproduct

	Key Partners	Key Activities	Key Resources	Costs	Value Propositions	Customer relationships	Channels	Customer Segments	Revenues
Subtraction									
Task Unification									
Multiplication									
Division									
Attribute dependency									

30.3 SCAMPER Idea Variation

- SCAMPER is a Solution process (see course ASICS)



http://de.wikipedia.org/wiki/SCAMPER

[Bob Eberle 1997]

- SCAMPER is a variation technique with 6 algebraic variation operators
- Derived from OSBORN checklist
- Kilbride's SCAMMPERR (SCAMPER+) adds two variation operations

S	Substitute (Vary)	Substitute some parts of the solution, resources, channels etc.
C	Combine	Combine partial solution elements to a more complete solution
A	Adapt	Change the solution or function
M	Modify	Scale, change an attribute of the solution
M	*Magnify*	*Change the size of the solution*
P	Put	Put to (find) another use
E	Eliminate (Subtract)	Remove, subtract, reduce to core
R	Reverse	Invert order
R	*Rearrange*	*Change order*

Ex.: SCAMMPERR with Sensor-Based Diapers

[Bob Eberle 1997]

- Remember the water-sensor-based diapers...

S	Substitute (Vary)	Substitute a part: Substute cable of sensor against wireless
C	Combine	Combine partial solution elements to a more complete solution: Second app to do social community analysis, taking the analytics of other parents into account
A	Adapt	Change the solution or function: Do a sensor-based diapers for elderly and handicapped people
M	Modify	Scale, change an attribute of the solution:
M	Magnify	Change the size of the solution: Make the wireless sensor smaller to be taken into the bladder; and use it for incontinent people
P	Put	Put to (find) another use
E	Eliminate	Remove, subtract, reduce to core: Let the sensor ring – no app
R	Reverse	Invert order
R	*Rearrange*	*Change order*

Matrix Analysis SCAMPERxBMC

- For this aspect-oriented canvas analysis, create a table (matrix), brainstorm on the crossproduct
- Exercise: do the same for VPC and PainCanvas

	Key Partners	Key Activities	Key Resources	Costs	Value Propositions	Customer relationships	Channels	Customer Segments	Revenues
Subtraction									
Combine									
Adapt									
Magnify/ Modify									
Put									
Rearrange/ Reverse									

Matrix Analysis SCAMPERxVPC

- For this aspect-oriented canvas analysis, create a table (matrix), brainstorm on the crossproduct
- Exercise: do the same for VPC and PainCanvas

	Customer Tasks	Gains	Pains	Pain Killers	Gain creators	Advantages	Features
Subtraction							
Combine							
Adapt							
Magnify/ Modify							
Put							
Rearrange/ Reverse							

30.4. Variability-Based Business with Rajkar's Idea Hexagon

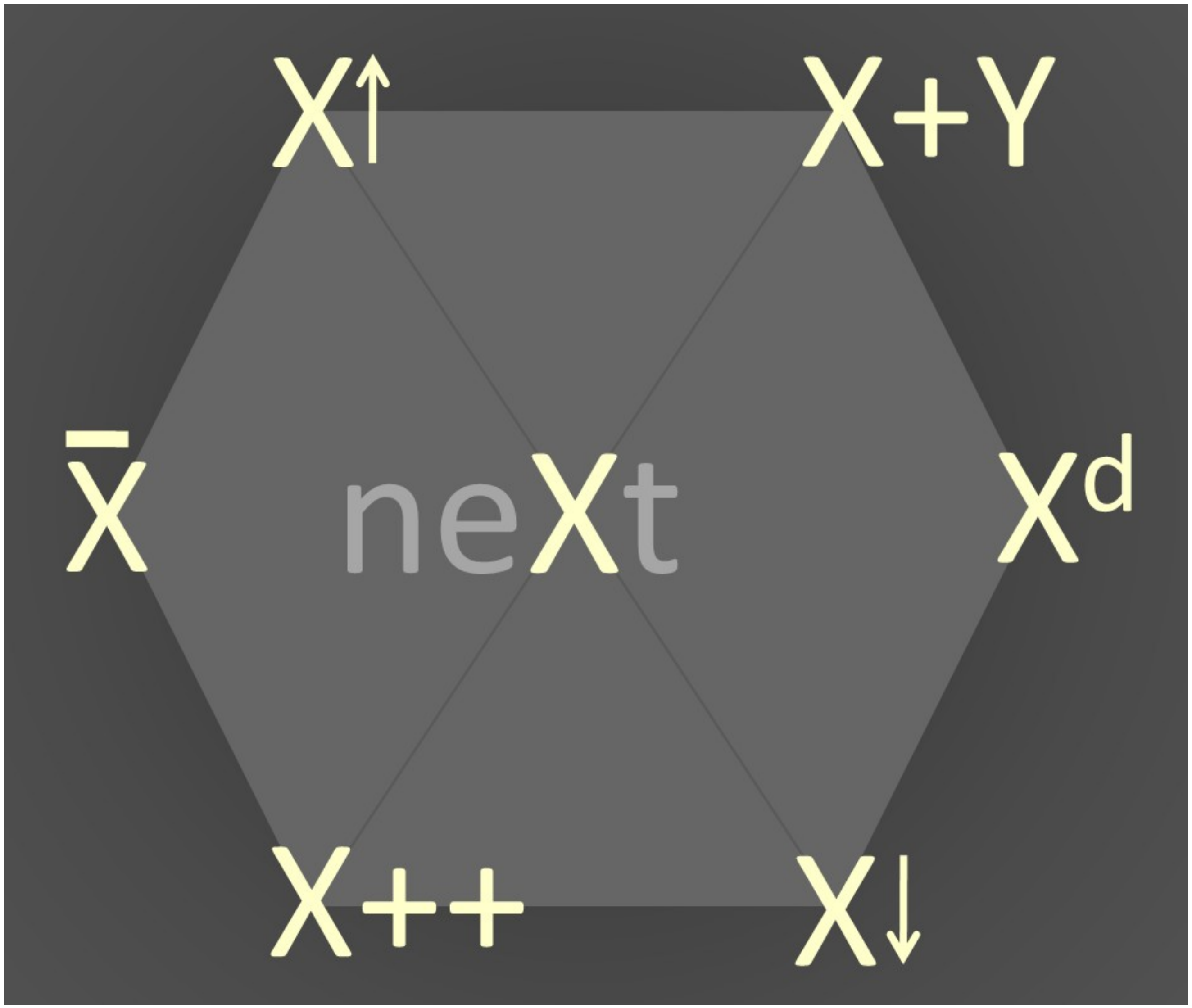
Slideshare Lecture of Rajkar / MIT.

The 6 Operations of Rajdar’s Idea Hexagon

[http://de.slideshare.net/cameraculture/raskar-ideahexagonapr2010]

- 6 Operations (DROHNF) to get new ideas (not only for products, also for technology)

D	Dimensional extension	Add another dimension to the idea
R	Restricting adjective	Add a new constraining adjective to the solution
O	Opposite	Do exactly the opposite
H	Find a hammer for a nail (abstracting, frameworking)	Search a new generic idea for an application; a new solution for a problem
N	Search for a new nail for the hammer (re-concretizing, framework re-instantiating)	Search a new application for a generic idea; a new problem for a solution
F	Fusion	Fuse dissimilar ideas into one idea



By Mituser2000 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=74796337

Idea Hexagon x VPC

- The operations are important for Product Line Engineering:
    - Dimensional extension (creates Product Matrices)
    - Hammer (creates frameworks)
    - Nail (instantiate frameworks)

	Customer Tasks	Gains	Pains	Pain Killers	Gain creators	Advantages	Features
Dimensional extension							
Restricting							
Opposite							
Find Hammer							
Find Nail							
Fuse							

30.5 SAMM

SAMM http://www.creapedia.com/w/index.php5/SAMM

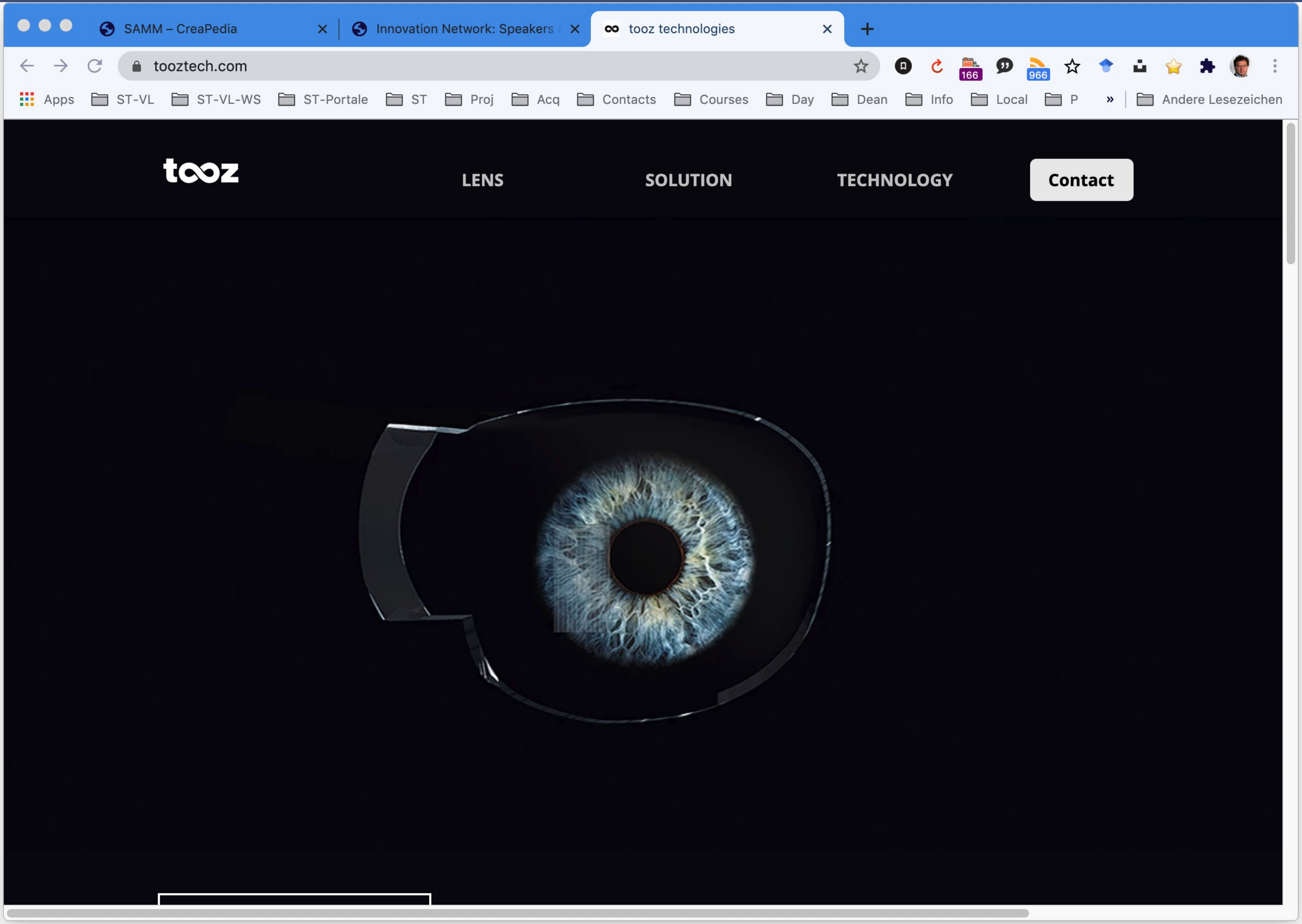
als SCAMPER+ für Aktivitäten

Ex.: SCAMPERR with Sensor-Based Diapers

- Think about the steps of a process

S	Substitute (Vary)	Substitute a process step
C	Combine	Combine several process steps to a macro-step
A	Accellerate	Faster...
M	Modify	Scale, change an attribute of the solution:
P	Prioritize	Prioritize process steps in importance
E	Eliminate	Remove a process step
R	Reverse	Invert order of process
R	Rearrange	Change order, parallelize

Tooz iGlasses



Ex.: SCAMPERR with Sensor-Based Diapers

- Refactor the steps of controlling the humidity of your child's diapers

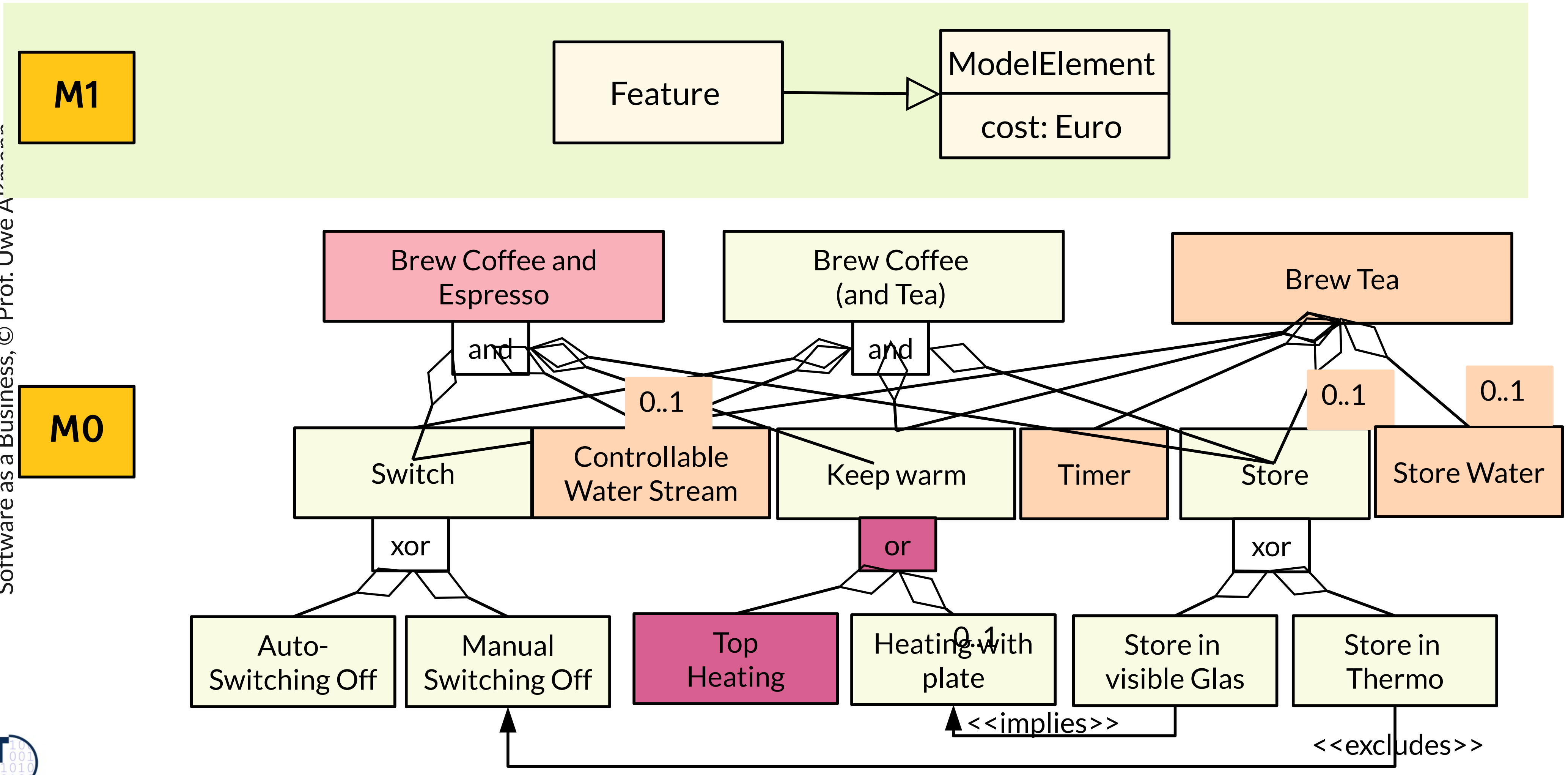
S	Substitute (Vary)	Substitute a process step: control humidity by app
C	Combine	Combine several process steps to a macro-step: check the 14-days analytics
A	Accellerate	Faster... Use a tooz glasses to blend in the status of your child's diapers into your eye
M	Modify	Scale, change an attribute of the solution: Vary humidity level (dry, semi-dry, wet, real-wet)
P	Prioritize	Prioritize process steps in importance: weigh humidity warning vs. humidity ignorance in groups
E	Eliminate	Remove a process step: eliminate manual intervention
R	Reverse	Invert order of process: not possible
R	Rearrange	Change order, parallelize. Let the app chose whether father or mother changes diapers

30.6. Incremental, Scalable Costs

The left side of the BMC talks about costs. How can we make them incremental and scalable?

Costs of Features in a Feature Model (of Product Line)

- Every feature node may have a ***cost attribute***
- Def.: The ***cost of a feature*** is the sum of the costs on the feature path
- Def.: The ***cost of a product*** with a set of features is the sum of all feature costs
- Def.: The ***cost of a product line*** is the cost of all nodes in its feature model

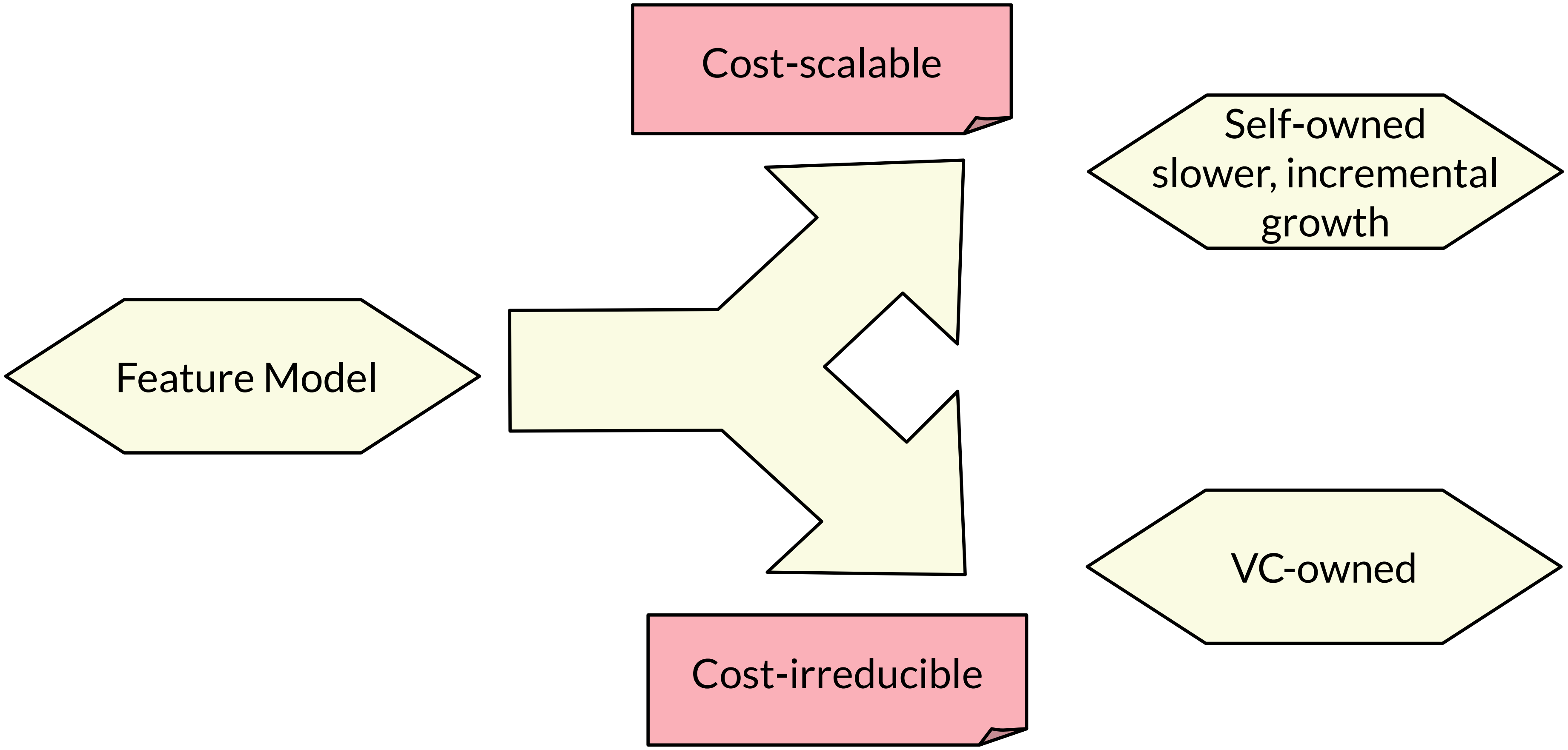


Cost Scalability and Venture Capital

- A company has a problem with an MVP if the cost of one feature is too high.
- It is better to have feature models with low costs per feature, because then new features can be added easily to the MVP
- Def.: A ***cost-scalable feature model*** is a feature model with low costs for every feature (i.e., all nodes).
- Def.: A ***cost-irreducible feature*** is a feature with high costs the company cannot bear alone, but needs capital partner

Law of Venture Capital (VC):
Startups with a cost-scalable feature model do not need VC.
Startups with a cost-irreducible feature need VC.

Two Ways for Building a Company



Ultimate Cost Advantage of an SPL (SPL-UCoA)

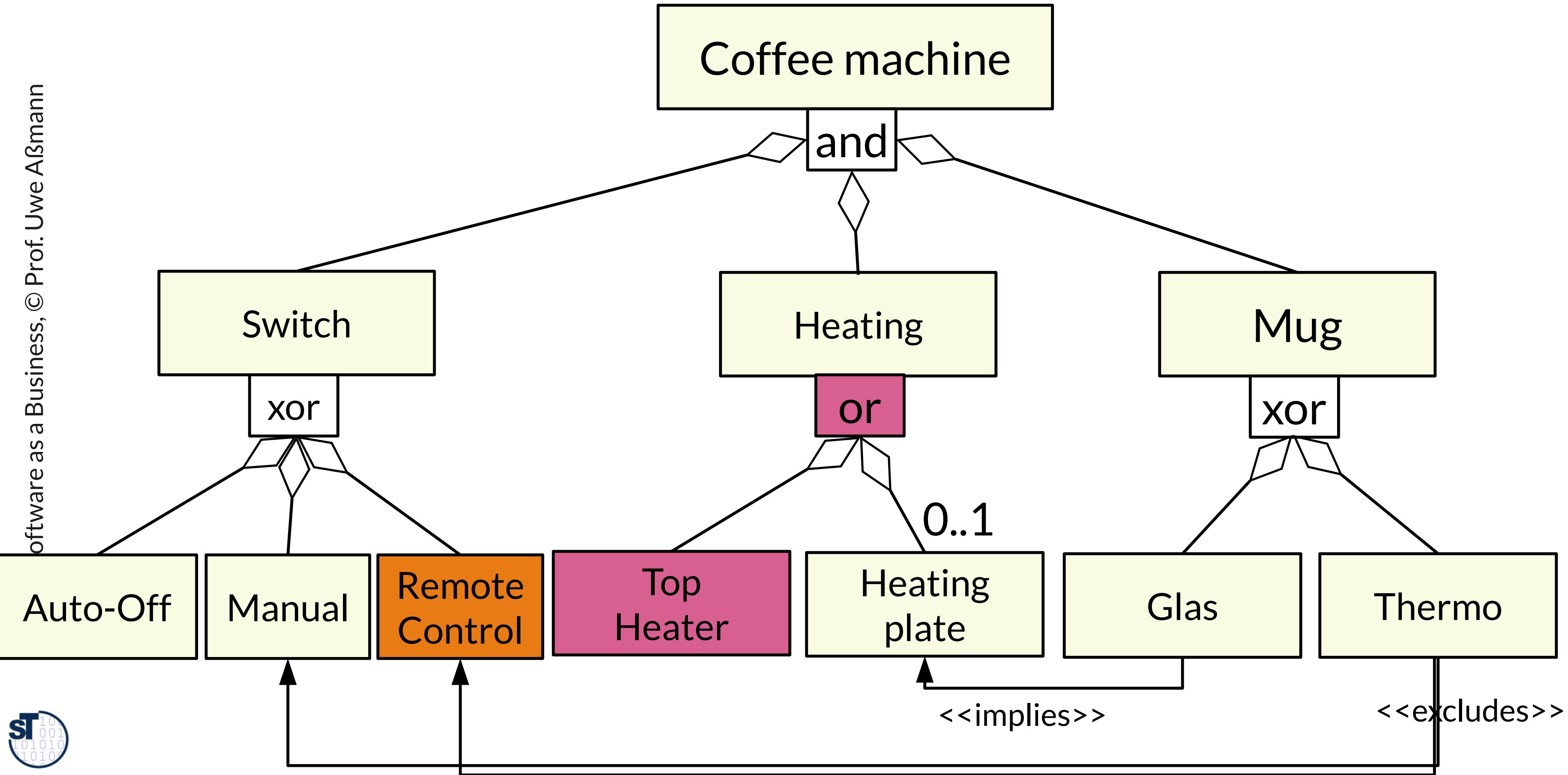
- Ultimate Cost Advantage (UCoA) is a UCA in the cost dimention.
- Make sure that all features in an SPL are extendable (scalable), and share subfeatures
- Then all products in the SPL add their value for the customer (and customer buys more products).

Law of SPL-UCoA:

Scale the products, but limit the costs by reuse of components and features.

Varianting Means to Add Alternatives to a Product Component Tree (2)

- Step by step, new components (for new features) can be added



No Idea for Scaling your Business?

- From an MVP, use an idea variation technique to arrive at:
  - Products with more features
  - Products with parameters
  - Feature and component outlining lead to product lines
  - Software ecosystems result if you allow third parties to do the idea variation, i.e., program their own apps
- Plan the scaling early on, but implement step by step
  - The first customers have to finance the next ones
  - Keep the IPR inhouse, only sell non-exclusive licenses to customers

The End

- What is the difference of a component tree and a feature tree?
- Why does the MVP focus on minimal viable *features* instead of minimal viable *components?*
- How do you extend a feature model of an MVFS with more alternative features? Give an overview of the major process steps.
- Explain Raijdar's Idea Hexagon and how to use it to generate new ideas.
- How do you use SCAMPER to get new product features?
- Explain the difference of SCAMPER and S.I.T.
- Suppose you have identified a MVFS, how to find more features?
- Why is it important to cross the 4 BMG operations with the BMC?
- Why is SAMM important for customer touchpoint analysis?