TECHNISCHE UNIVERSITÄT DRESDEN

31. Maintaining a Product Family of Canvases

Lean (Canvas) Modeling -
Grading and Metrics on Canvases

Prof. Dr. Uwe Aßmann
Technische Universität Dresden
Software Engineering Group
http://st.inf.tu-dresden.de

Version 19-0.4, 07.12.19

1) Canvases as collaborative tools
2) Lean modeling with canvases
3) Nested canvases
4) Grading and metrics on canvases
5) The canvas cactus as megamodel
6) The canvas product family

Literature

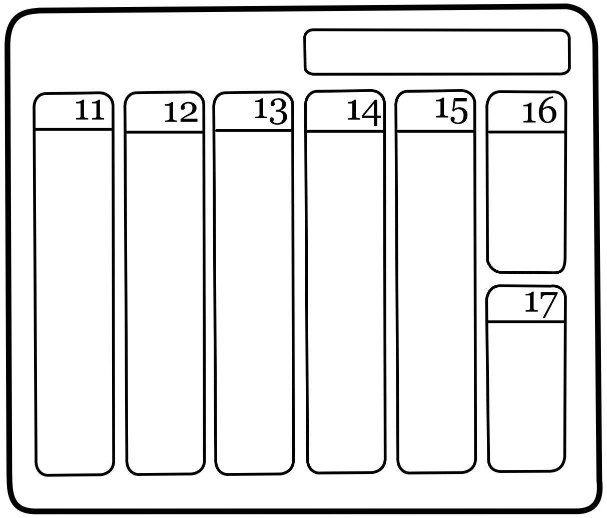
[CM03] Sitt Sen Chok, Kim Marriott. Automatic Generation of Intelligent Diagram Editors. ACM Transactions on Computer-Human Interaction, Vol. 10, No. 3, September 2003, Pages 244–276.

31.1 Canvases as Light-Weight Cooperation Tools

- We can vary a feature model with idea variation techniques.
- Thereby, feature models grow; and hopefully later more products grow your business.
- Here, we learn, how to link canvases to the growing feature models.

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

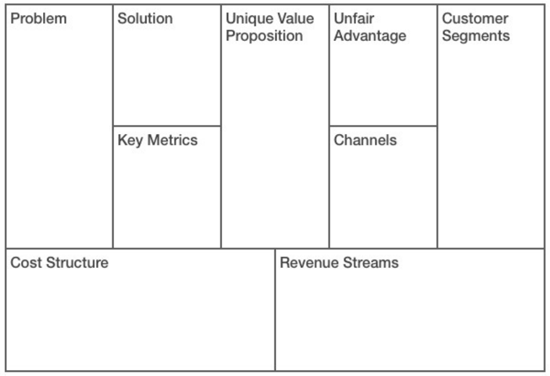
Canvases as Lean Models

- A **canvas** is a collaborative frontend for a model, in which sticky notes demarcate the formal content from the informal text.
- A **lean model** is a *semi-conceptualized model*, an active document with informal and conceptualized content, fulfilling *some constraints* of a set of constraints.
  - Models fulfill all of them.
- **Lean modeling is an agile conceptualization process:**
  - Canvas -> Lean Model -> fully conceptualized Model

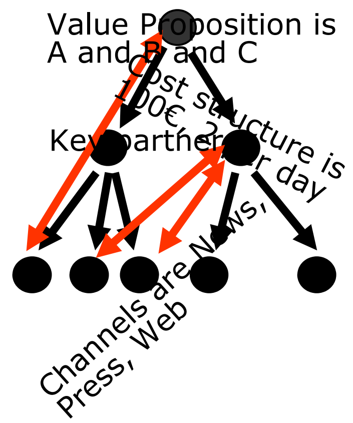
Canvas

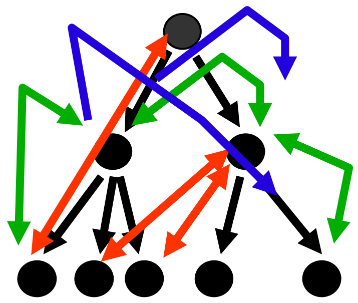
Lean Model
(semi-conceptualized, some constraints)

Model
(all constraints)



Lean Canvas is adapted from The Business Model Canvas (http://www.businessmodelgeneration.com) and is licensed under the Creative Commons Attribution-Share Alike 3.0 Un-ported License.





31.2. Lean Modeling with Canvases

Schemas for Flat Canvases as Grammars

- A **(flat) canvas** is a structured questionnaire for collaborative development
- It can be represented as a **tree-shaped model with constraints**
    - Canvas structure:
        - Canvas left side vs. right side
        - Left part, right part, upper, lower part
        - Canvas fields with sticky text notes, Canvas questions or answers
    - Inter-field references with inter-field constraints
    - Intra-field constraints
    - Canvas fill order (partial order) on the tree nodes
    - NO Subcanvases; Subcanvases are other trees that may be referenced

Schemas for Flat Canvases as Grammars

- The canvas' schema can be *described by a grammar in a Part Grammar (Constraint Multiset Grammar*, *CMG)* describing Whole-and-Part relationships. **Example invariants**:
  - forall stickynotes in CustomerRelations there is a stickynote in Channels;
  - there **must** be a revenue;
- Why is the partial fill order a set of inter-field constraints?
  - Alternative language for grammars and constraints: EBNF and OCL

```
// Example Grammar for BMC
Grammar Fields = { PartRules {
  Note ::= Question | Answer
  Root Field ::= StickyNote:Note *
}}
Grammar BusinessModelCanvas = {  import Fields
 PartRules {
  Root BMC ::= { LeftPart ValueProposition:Field RightPart }
  LeftPart ::=  { KeyPartners:Field KeyActivities:Field KeyResources:Field Costs:Field }
  RightPart ::= { CustomerRelations:Field Channels:Field CustomerSegments:Field Revenues:Field }
 }  Invariants {
  Invariant { forall s:CustomerRelations.StickyNote* exists y:StickyNote, y in Channels.StickyNote*
  Invariant MUST exists r:StickyNote in Revenues.StickyNote* }
 }
}
```

VPC as Grammar with Constraints

```
Grammar ValuePropositionCanvas = import Fields {
 PartRules {
  Root VPC       ::= { LeftPart RightPart }
  LeftPart ::= { GainCreator:Field PainKiller:Field ProductsAndServices:Field }
  RightPart      ::= { Gain:Field Pain:Field CustomerSituation:Field }
 } Invariants {
  Invariant forall s:Gain.StickyNote* exists y:StickyNote, y in GainCreator.StickyNote*
  Invariant forall s:Pain.StickyNote* exists y:StickyNote, y in PainKiller.StickyNote*
  Invariant forall s:PainKiller.StickyNote* exists y:StickyNote, y in ProductsAndServices.StickyNote*
  Invariant forall s:GainCreator.StickyNote* exists y:StickyNote, y in ProductsAndServices.StickyNote*
 }
}
```

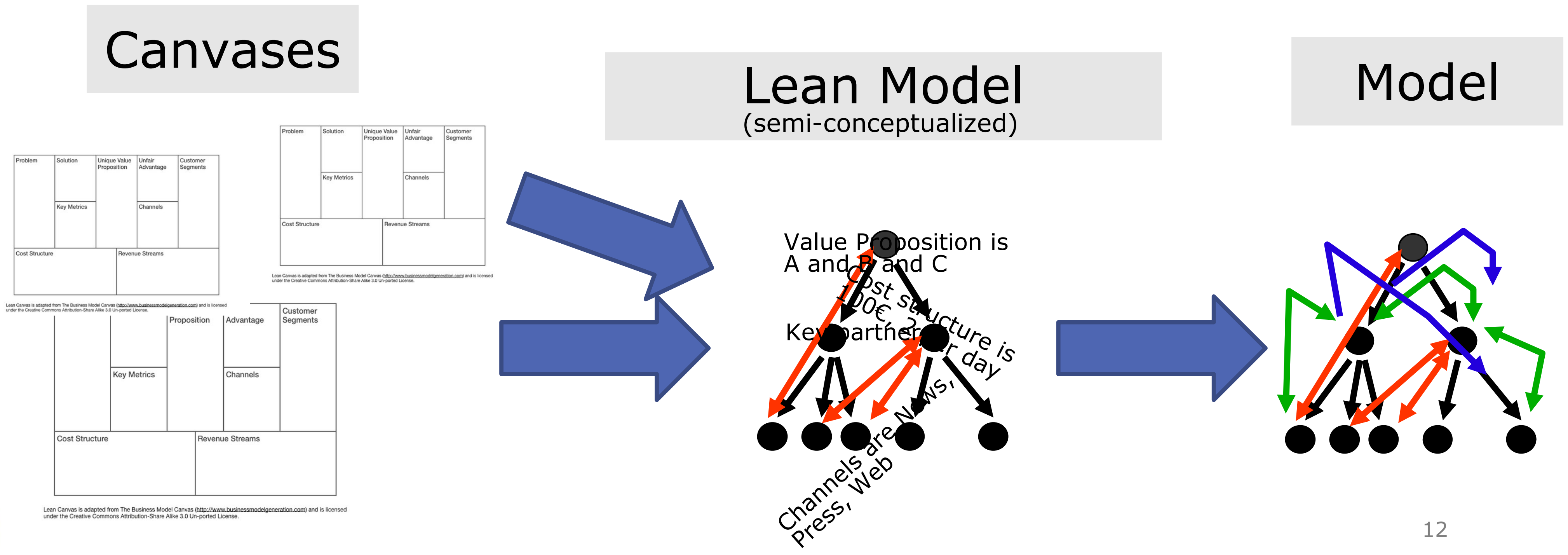
- Invariants:
  - Forall gains there must be a gain creator
  - Forall pains there must be a pain killer
  - Forall pain killers there should be a service or product
  - Forall gain creators there should be a service or product

Validating a Flat Canvas

- A flat canvas is called **well-formed**, if
    - All fields are being computed (filled)
    - All fields fulfill all constraints.
- Validation:
    - Parse the canvas with its sticky notes
    - Evaluate constraints in OCL
    - or with an Attributed Grammar
    - or with an Multiset Constraint Grammar

Parallelly Edited Lean Models can be Merged

- A **lean model** can be merged with another lean model
- A **canvas twin** is a parallelly edited canvas, which can be merged into a lean model by unifying the fields
- **Conceptualization Process:**
  - CanvasTwin * -> Lean Model -> fully conceptualized Model
  - Assembling all constraints
  - Validating all constraints


Canvases
Problem
Solution
Unique Value Proposition
Unfair Advantage
Customer Segments
Key Metrics
Channels
Cost Structure
Revenue Streams
Lean Canvas is adapted from The Business Model Canvas (http://www.businessmodelgeneration.com) and is licensed under the Creative Commons Attribution-Share Alike 3.0 Un-ported License.
Lean Model
(semi-conceptualized)
Value Proposition is A and B and C
Cost structure is 100€ / 2 per day
Key partner
Channels are News, Press, Web
Model

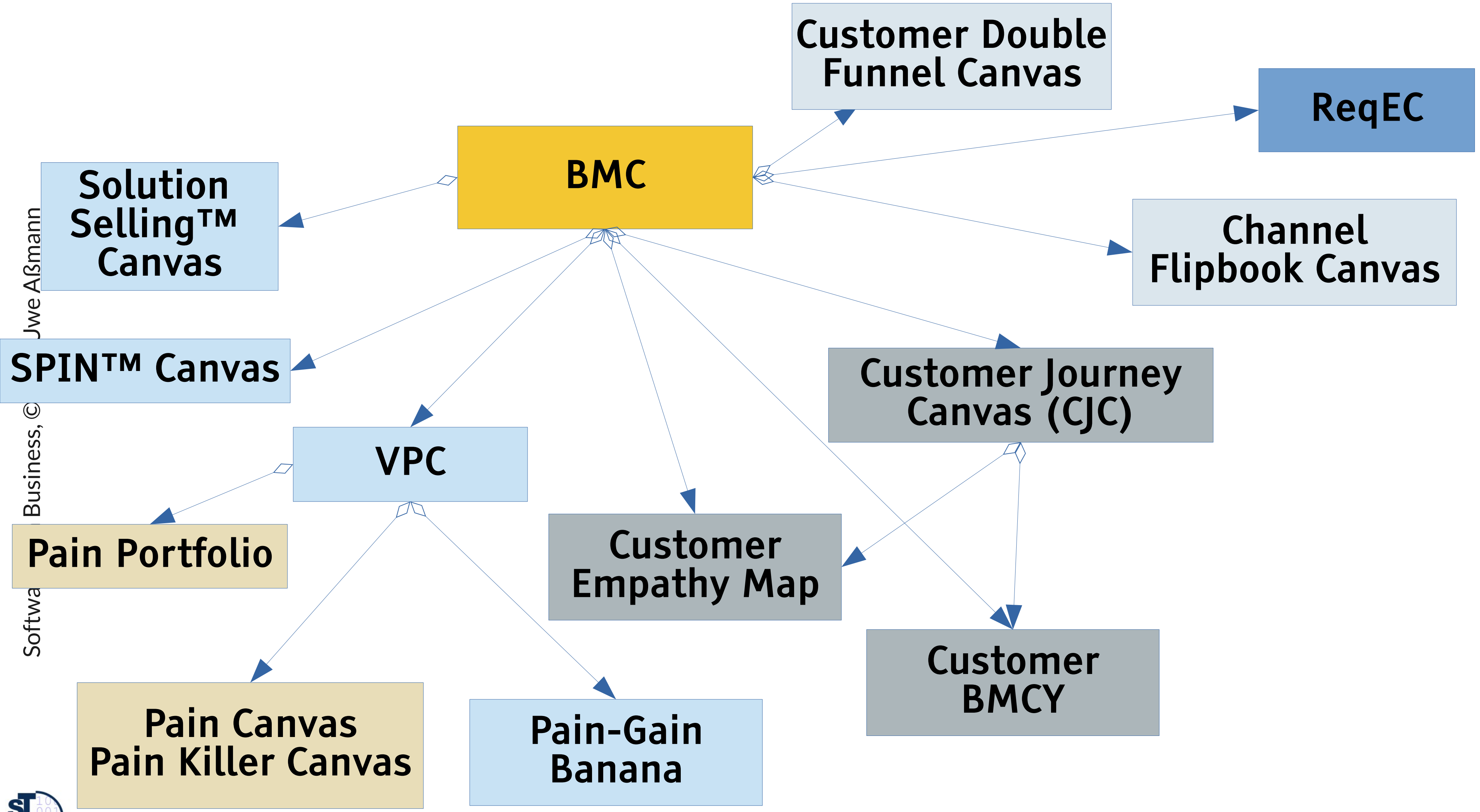
31.3 Nested Canvases

A Nested Canvas

- A ***nested canvas (deep canvas)*** is a link tree with different levels
  - Every canvas forms a sequence, graph or array of fields
  - Sticky notes attach text to the fields
  - Constraints constrain the content of the canvas fields
- **Subcanvases** form children
  - Grammars of nested canvases are united (grammar composition)
- The **fill order** of the canvas defines a phase structure on the link tree
  - Metrics on advancement (hierarchical wavefront progress)

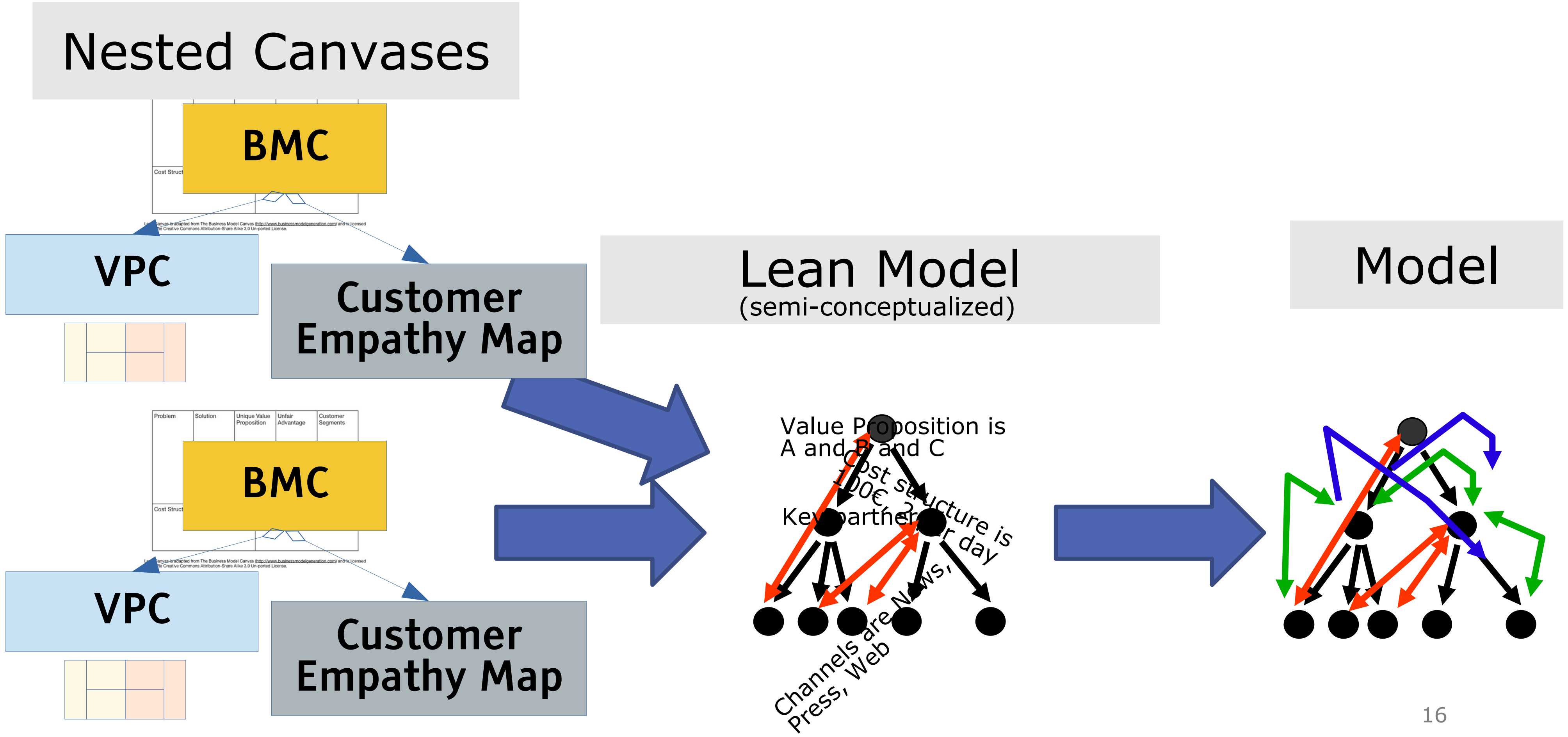
The Nested BMC (Deep BMC)

- Many subcanvases



Parallelly Edited Lean Models can be Merged to Get a More Mature Model

- A **nested canvas twin** is a parallelly edited nested canvas, which can be merged into a lean model by unifying the fields
- **Conceptualization Process:**
  - NestedCanvasTwin * -> Lean Model -> fully conceptualized Model



31.4 Grading and Metrics on Canvases

Assessment in Canvases and Nodetypes in Canvas Trees

- **StickyNote dimension:** every node can have a sticky note (Answer to a canvas question)
- **Commenting** is done by spanning up a ***comment dimension*** in a canvas tree
  - Every node can get a comment
- **Corresponding dimension:** Every node (e.g., sticky note or comment) can invoke a corresponding node in another field that has to be filled
  - When a sticky note invokes another sticky note
  - INVARIANT Exists s:StickyNote: corresponding(self, s)
- **Grading** is done by spanning up a *grading dimension* in a canvas tree
  - Every node can get a grade (green-yellow-red, 1-5, 1-10, 1-15)
  - The grading dimension defines grading functions for sticky notes in the fields
- **SWOT dimension:** every node can get a SWOT grading node: "how strong/weak/opportunity-like/trend-like is node?"
  - BMC-SWOT grading matrix canvas uses the SWOT grading dimension
  - LeanCanvas-SWOT uses SWOT grading dimension for LeanCanvas
- **Grading on nested canvases:** Grading is like commenting, but attributing a grade to a node. It defines the grading functions for all tree nodes of the nested canvas.

Examples of Attributes (Variables) of a Canvas Field (Node)

- Node.Questions: List(Question) *// all questions of a field or note*
- Node.SWOT: List(SWOT)
- Node.Comments: List(Comment) *// all nodes in a canvas can be commented*
    - NumberOf *// all lists in nodes of a canvas can be counted*
- Field.AllStickyNotes: List(StickyNotes)
- Field.MissingStickyNodes: List(empty Fields)
- Field.Grade: */* The average of all sticky note grades */*
- StickyNote.Grade: */* the grading: e.g., red, yellow, green */*
- StickyNode.SWOT.Strength.Grade: */* Grade of SWOT */*
- StickyNode.SWOT.Weakness.Grade: */* Grade of SWOT */*
- StickyNode.SWOT.Opportunity.Grade: */* Grade of SWOT */*
- StickyNode.SWOT.Trend.Grade: */* Grade of SWOT */*
- StickyNote.CorrespondingStickyNote: List(Ref StickyNote) */* corresponding sticky nodes or holes */*
- Canvas.Grade: */* The average of all sticky note grades of all nodes */*

Thresholds for Canvas Metrics

- Status of invariants is important for the ***maturity* of the canvas**

A **green** canvas fills all its variables
and fulfills all its invariants.

Only green canvases are models.

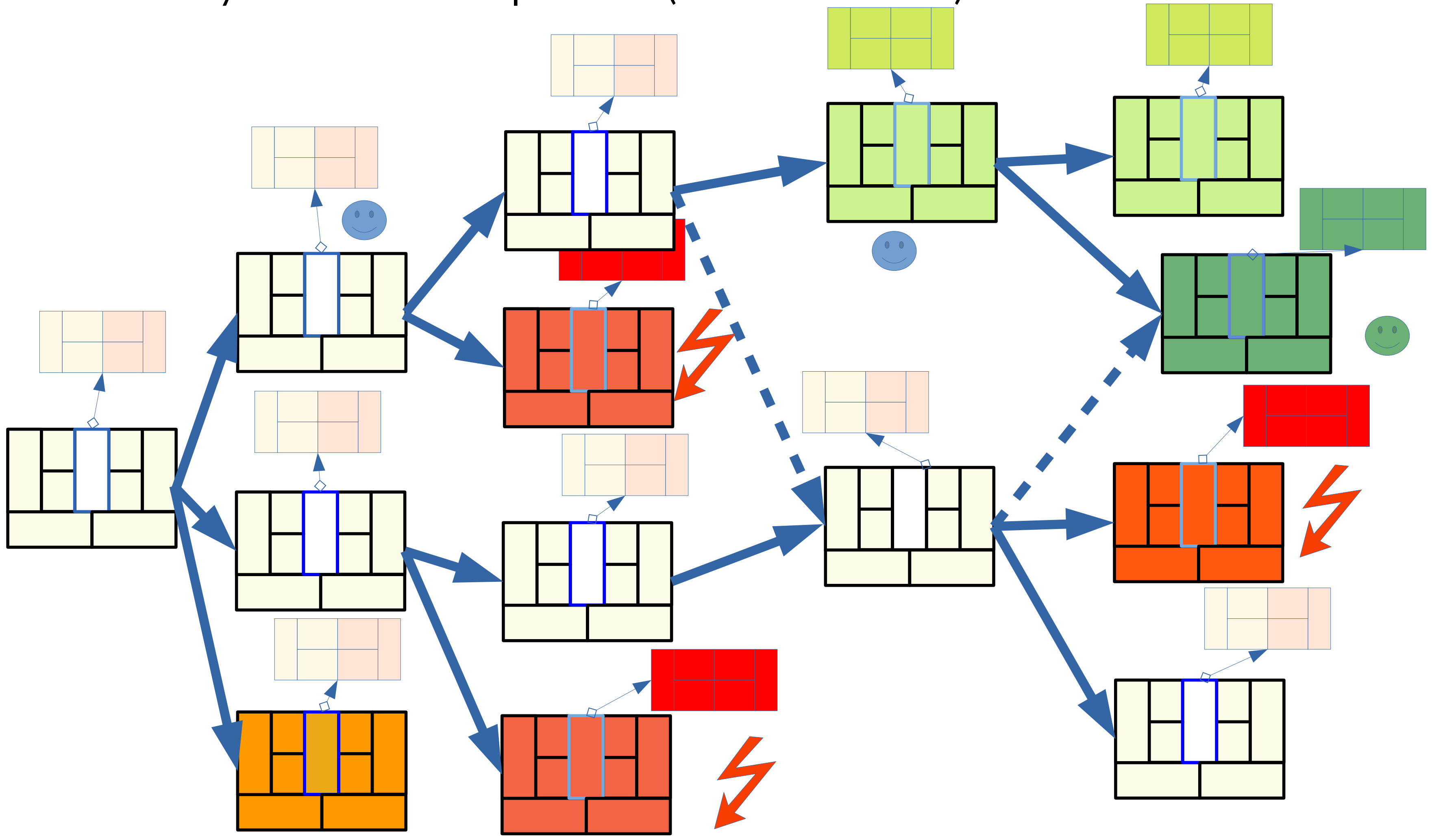
If a set of metric function on a nested canvas does not
fulfil its threshold, or if not all invariants are fulfilled,
we call the canvas ***orange.***

A **red** canvas does not fulfill all its **MUST invariants.**

31.5 The Canvas Cactus as Megamodel and its Metrics

The Evolving BMC-VPC Canvas Cactus (extended)

- Growing a tree with side edges (link tree - cactus) out of a first version
    - Assess with red-yellow-green; choose a current “greenest” “champion”
- Every step tests **hypotheses** about the customer
- Not too many canvases are kept active (small dashboard)

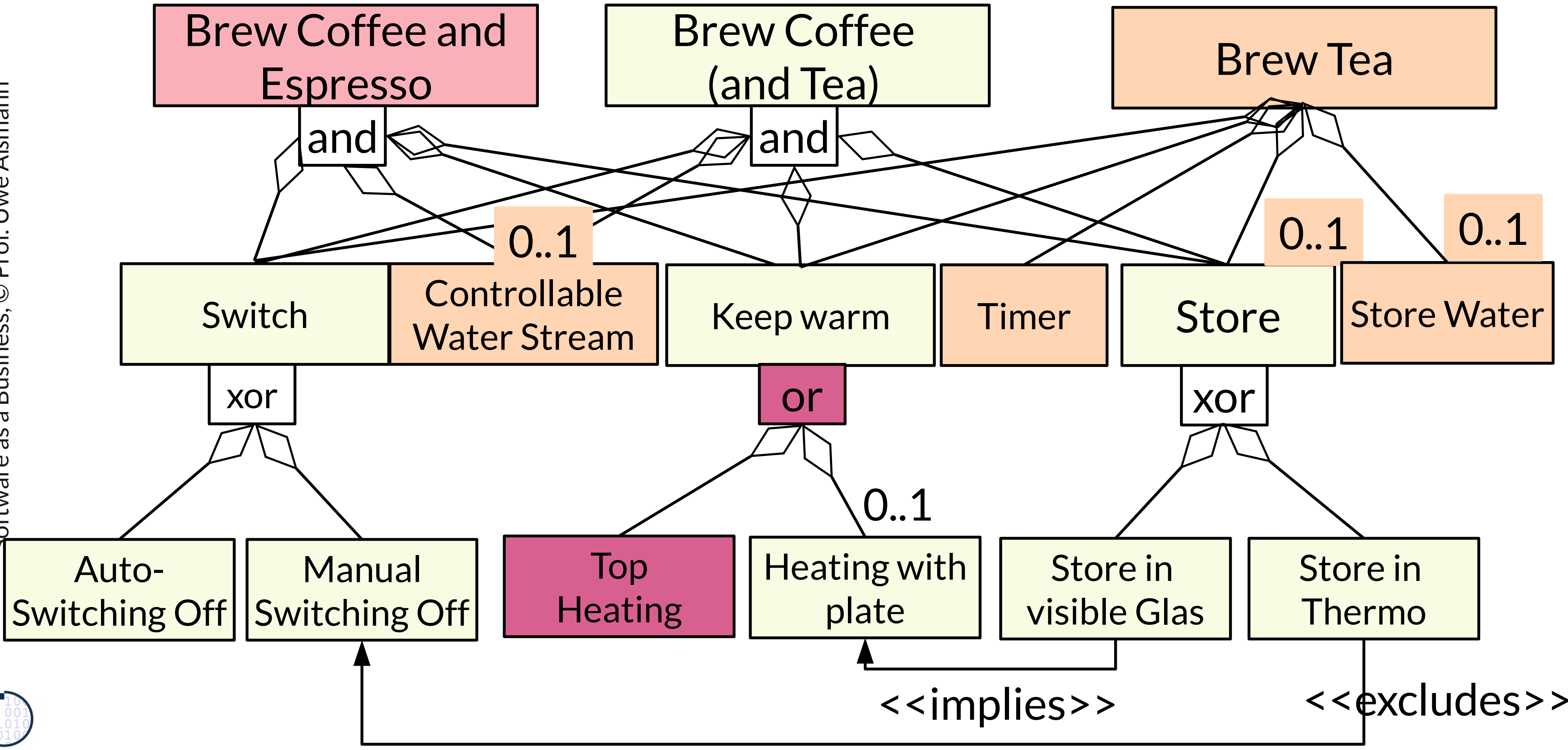


The Megamodel of Evolving Canvases

- A **megamodel** describes a set of models
- Find new ideas with idea variation techniques
- A **canvas cactus** is a link tree of canvases, i.e., a link-tree-shaped megamodel of canvases
- Canvas cactus evolution evolves the megamodel with agile modeling
- The megamodel of canvases in a cactus is a link tree and can be analysed by constrained multiset grammar (CMG)
    - Metrics
    - Constraints

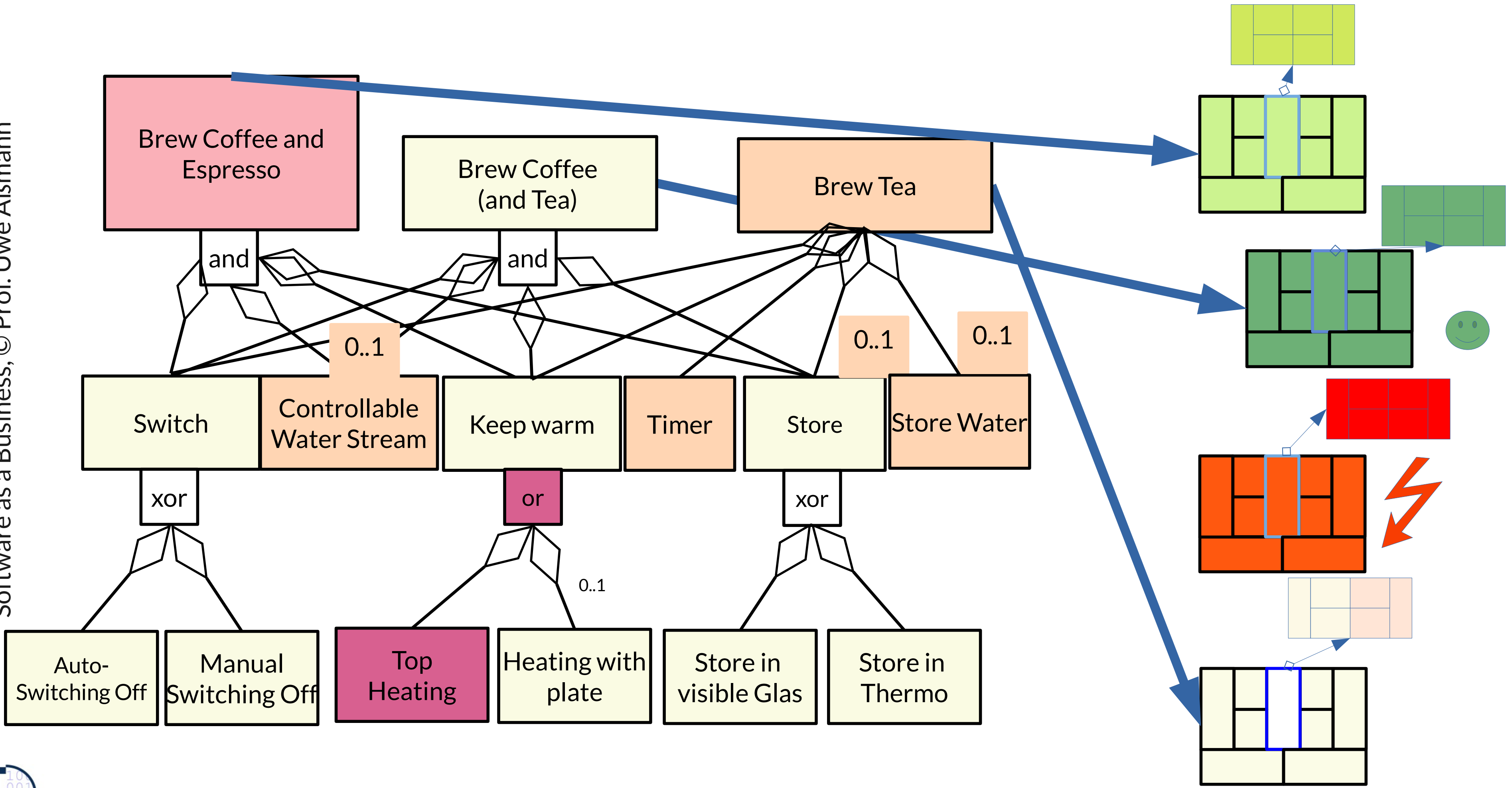
Remember: Extended Feature Model (of Product Line)

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



The Product Line of Canvases

- Combined with a feature model, the green canvases document all products of your product line
- A **canvas family** is a feature model linked with the corresponding canvases



The End

- More on modeling, lean modeling, and megamodeling in the course
- “Model-Driven Software Development in Technical Spaces (MOST)” in WS

- Explain the concept of a CMG. Why do we need a grammar to model Canvases?
- Explain why a canvas is an instance of a CMG.
    - Which role do invariants play?
    - Which role do filling functions play?
    - Can the user execute / simulate a filling function?