Feature Modeling

Krzysztof Czarnecki & Simon Helsen
University of Waterloo
czarnecki@acm.org

www.generative-programming.org

Overview

- ➢ Basic Feature Modeling Concepts
- Exercise
- AmiEddi-XML 1.3
- Captain Feature
- Homework

Feature Modeling

- Feature modeling is about identifying
  - common features of concepts
  - variable features of concepts
  - and their dependencies

  and documenting them in a coherent model, called a feature model
- Feature modeling is a core activity of important domain-engineering methods
- Features help to identify implementation components and designing DSLs

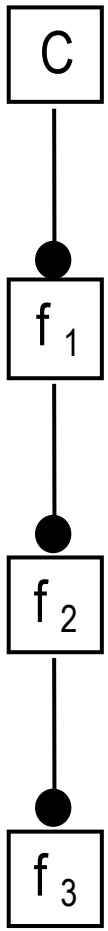
Features and Psychology

- The Human mind processes lots of information
- Important function of human information processing: reducing the amount of information
  - Describing concepts by features, e.g. a bird has wings, a bill, and can fly
  - Classifying a perceptional unit as an instance of a concept, e.g. Aunt Martha's bird
- Acquisition and evolution of concepts will not be pursued further in this course

Concepts and Classes

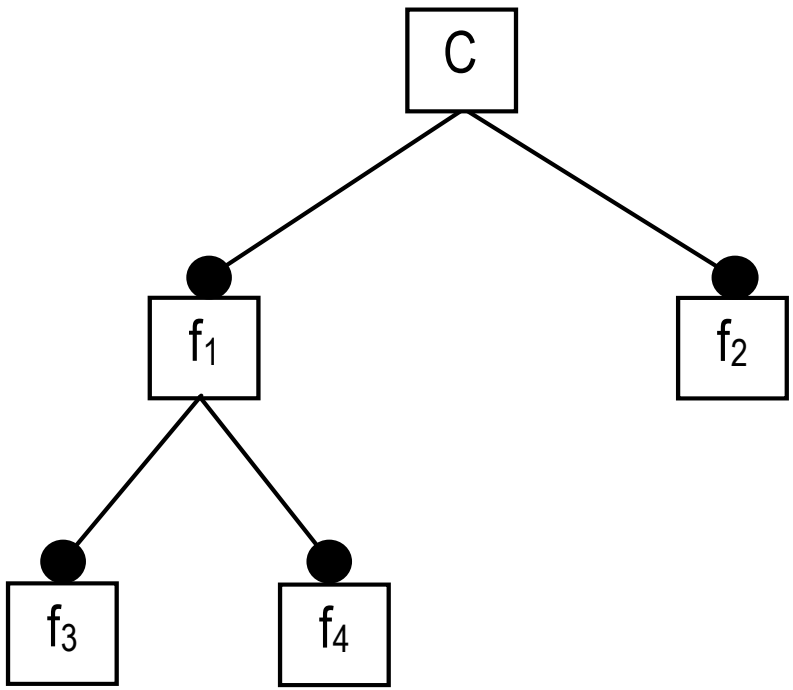
- Aren't these equations generally true?
  - Class = concept
  - Object = instance of a concept
- No!
  - Objects have predefined semantic properties such as identity, state, behavior
  - Instances of concepts do not have any predefined semantics, they could be virtually anything

Feature Diagram



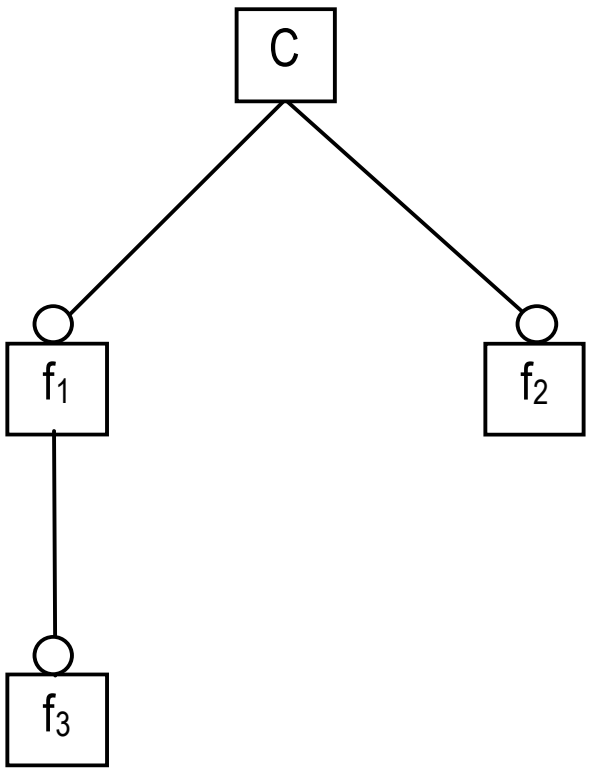
- The root node C represents the concept C
- All other nodes are features
- f1 is a direct feature of C
- f2 and f3 are indirect features of C
- f2 is a direct sub-feature of f1
- f3 is an indirect sub-feature of f1

Mandatory Feature



- A mandatory feature is part of a concept instance description only if its parent is also part of the description
- Mandatory features are pointed to by edges with a filled circle, e.g. f1,f2,f3, and f4
- All instances of C are described by the feature set {C,f1,f2,f3,f4}

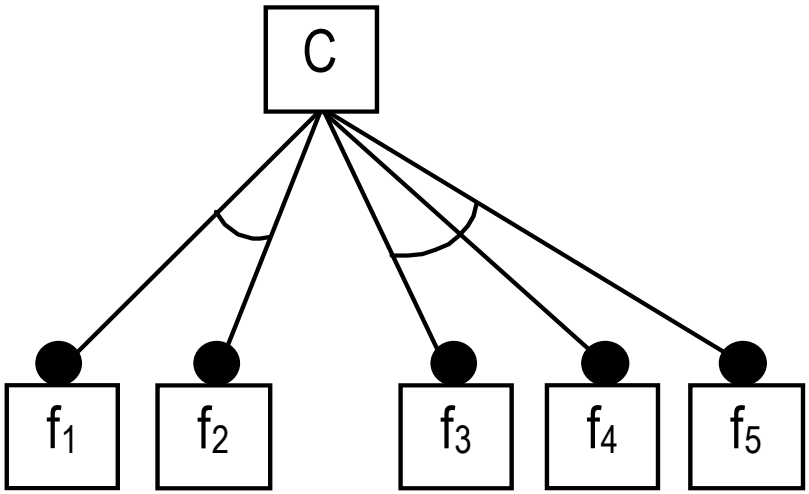
Optional Feature



- An optional feature can be part of a concept instance description only if the parent node is also part of the description
- Optional features are pointed to by edges with an empty circle (e.g., f1,f2, and f3)
- The following sets describe instances of C: {C}, {C,f1}, {C,f1,f3}, {C,f2}, {C,f1,f2}, {C,f1,f3,f2}

Exclusive-Or Features

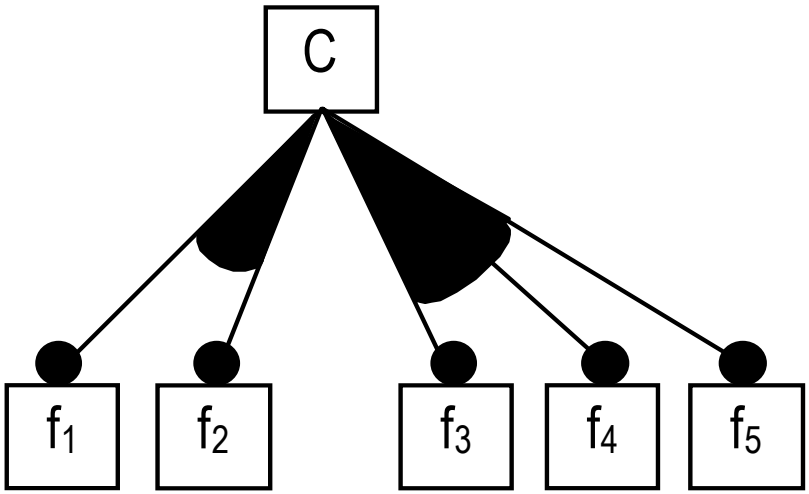
(In the textbook, these are called 'alternative-features')



- Exactly one from a set of exclusive-or features is part of a concept instance description if its parent node is also part of the description
- Edges pointing to exclusive-or features of one set are connected by an empty arc
- The following sets describe instances of C:
  {C,f1,f3}, {C,f1,f4},
  {C,f1,f5}, {C,f2,f3},
  {C,f2,f4}, {C,f2,f5}

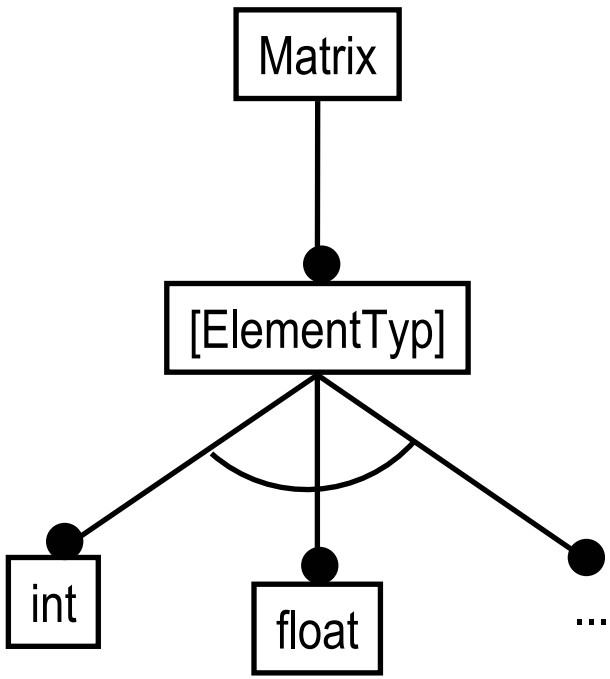
Inclusive-Or Features

(In the textbook, these are called 'or-features')



- Any non-empty subset from a set of inclusive-or features can be part of a concept instance description if the parent node is also part of it
- Edges pointing to inclusive-or features of one set are connected by a filled arc
- The diagram denotes ((2*2) – 1) * ((2 * 2 * 2) -1) = 21 different concept instances

Open/Closed Features

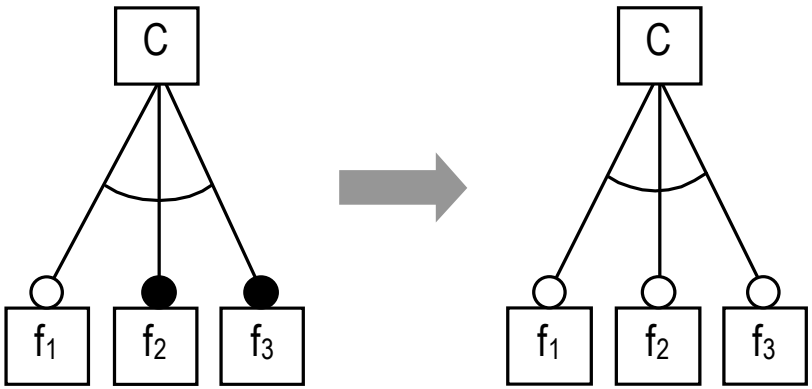


- An open feature is expected to be refined with further sub-features
- In a feature diagram, brackets [] are used to indicate openness
- We can also show selected examples of sub-features (not part of the formal notation)

Normalized Feature Diagrams

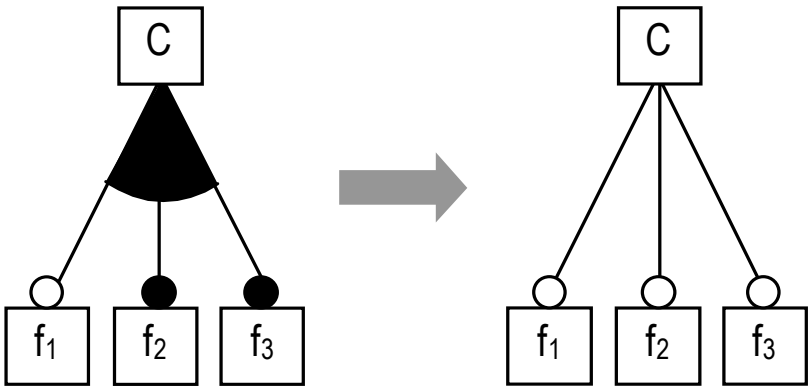
- Besides mandatory, optional, exclusive-or, and inclusive-or features, there are also
  - *exclusive-or-optional* features and
  - *inclusive-or-optional* features
- This results in redundant representations
- Each feature diagram can be transformed such that
  - any set of exclusive-or features is either fully exclusive-or or fully exclusive-or-optional
  - it does not contain any inclusive-or-optional features

Normalized Feature Diagrams



- If one or more of the features in a set of exclusive-or features is optional, it has the same effect as if all the exclusive-or features in this set were optional

Normalized Feature Diagrams



- If at least one or more of the features in a set of inclusive-or features is optional, it has the same effect as if all the features in this set were optional features

Commonality in Feature Diagrams

- From the perspective of a concept
  - What are the common features of all concept instances?

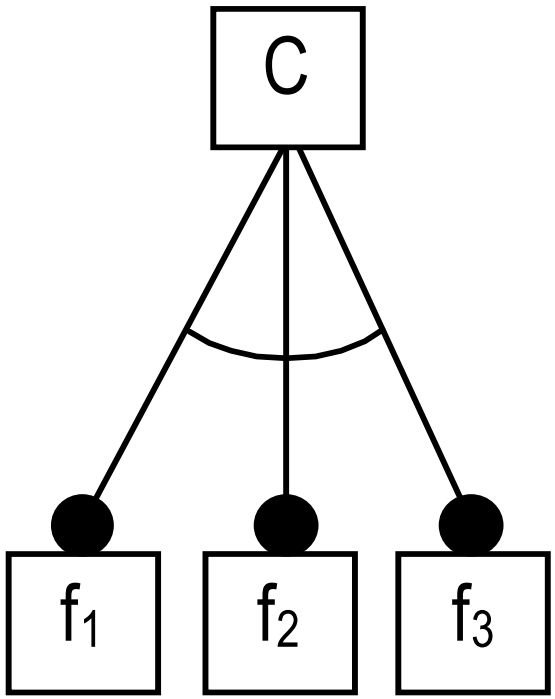
  ➔Common Features: all mandatory features of a concept of mandatory sub-features of common features
- From the perspective of a feature
  - What are the common features of all concept instances that have the special feature?

  ➔Common Sub-features: all mandatory sub-features of a feature or mandatory sub-features of common sub-features

Variability in Feature Diagrams

- Variable features describe variability
  - optional, exclusive-or, exclusive-or-optional, and inclusive-or features
- The nodes having variable sub-features are so-called variation points (VP)
  - homogeneous vs. inhomogeneous VPs: all direct sub-features are of the same vs. different kinds
  - singular vs. nonsingular VPs: none or one vs. many direct sub-features can be selected

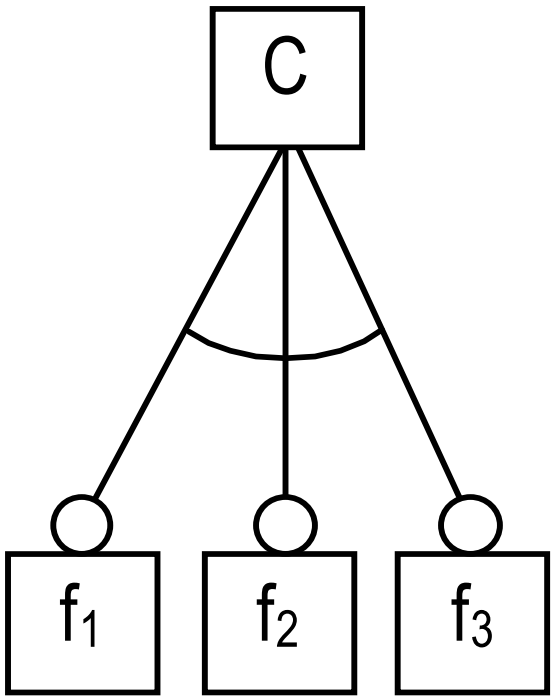
Kinds of Variation Points



Dimension

- Feature or concept of which all direct sub-features (or features) are exclusive-or features

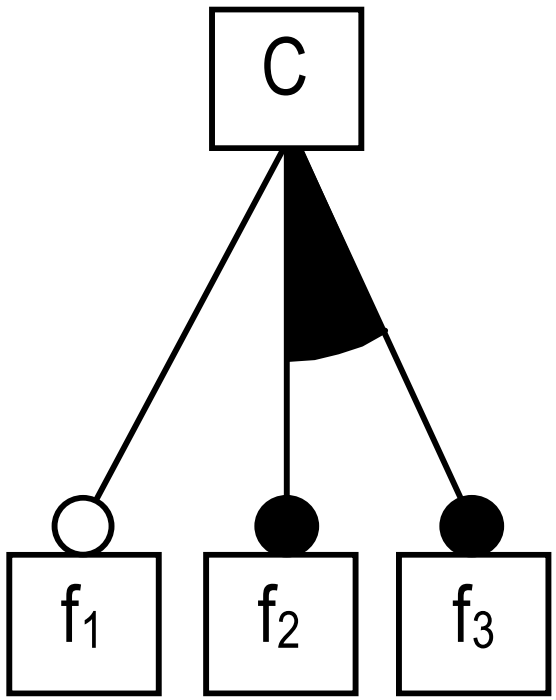
Kinds of Variation Points



Dimension with optional features

- Feature or concept of which all direct sub-features are alternative optional features

Kinds of Variation Points



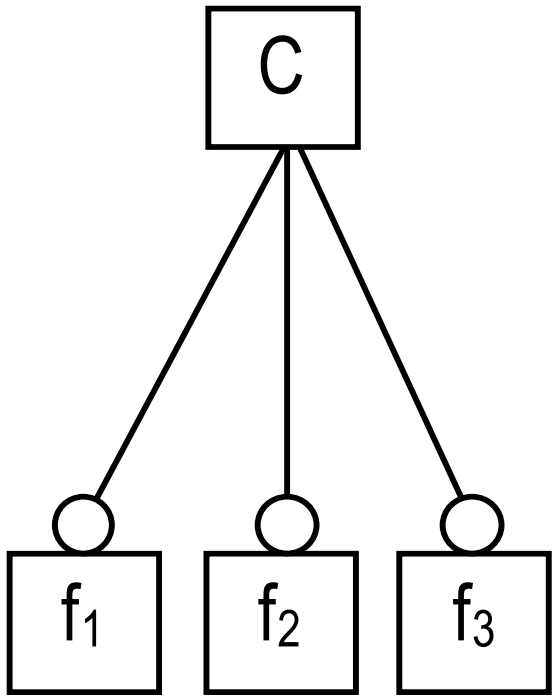
Extension point

- Concept or feature with at least one set of direct inclusive-or features or inclusive-or sub-feature

or

- Concept or feature with at least one direct optional feature or sub-feature

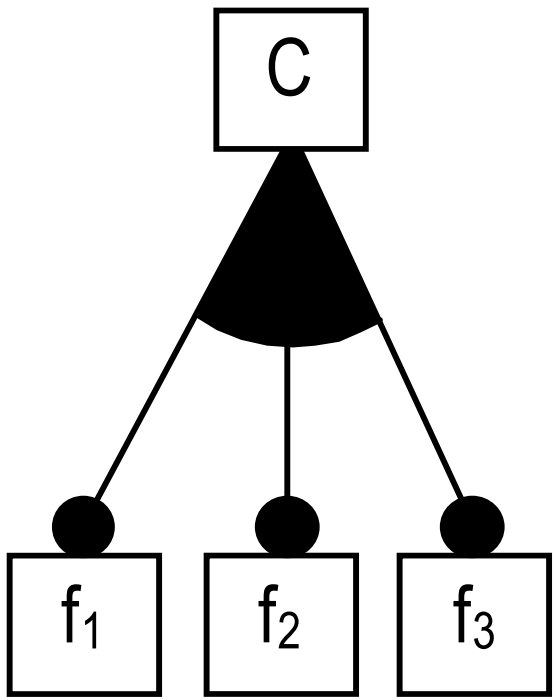
Kinds of Variation Points



Extension point with optional features

- Concept or feature with all direct features or sub-features being optional features only

Kinds of Variation Points



Extension point with inclusive-or features

- Concept or feature with all direct features or sub-features being inclusive-or features only

Feature Model Consists of

- Feature diagram(s) and
- additional information
  - Short semantic description of each feature
  - Rationale for the relevance of each feature
  - Stakeholders and systems that are interested in a feature
  - Examples of systems with that feature
  - Constraints
  - Defaulting dependency rules
  - Availability (where, when, and to whom is the feature available?)
  - Binding mode (dynamic, static)
  - Priority (how important is the feature?)
  - ...

FAQs

Is a feature diagram not just a simple part-of hierarchy?

- No, features can represent part-of relations, but they can carry any other meaning too
- Features can be implemented quite differently, e.g.
  - concrete features as concrete components
  - aspect features, e.g. synchronization, are often woven into other components
  - abstract features, such as memory usage optimization, determine the configuration of other components

FAQs

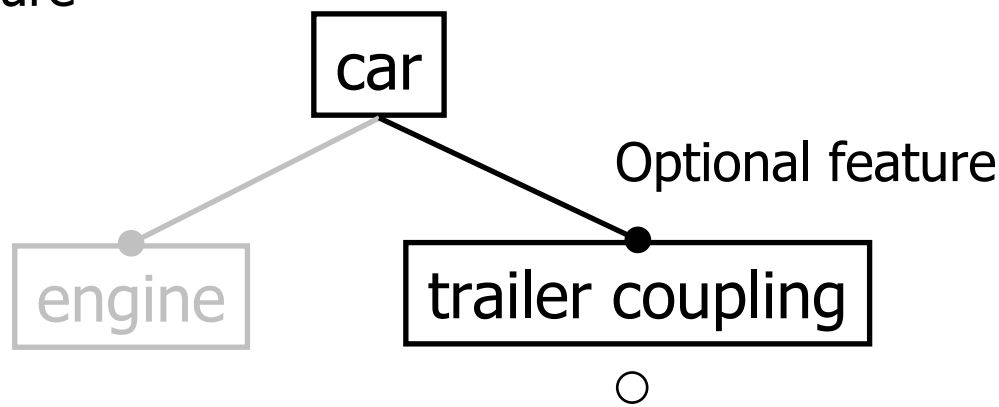
What are the semantics of the edges of a feature diagram?

- Edges do not have their own full semantics
- The decoration of the edges completes the semantics: a certain feature must, can, or can not be part of a concept instance description
- Edges do not represent part-of or inheritance relationships!

FAQs

How to describe the configurability aspect of a relation if edges do not possess relationship semantics?

- If a feature diagram contains a variable relation it should be modeled as a feature
- For example, the option ”trailer coupling" is represented as an optional feature



Feature Combinations

$$V(f) = \begin{cases} V'(f)+1 & if \quad optional(f) \\ V'(f) & otherwise \end{cases}$$

$$V'(f) = \begin{cases} 1 & if \quad atomic(f) \\ \left(\prod_{s \in mandoptsub(f)} V(s)\right) * \left(\prod_{g \in subgroup(f)} VG(g)\right) & otherwise \end{cases}$$

$$VG(g) = \begin{cases} \sum_{f \in g} V(f) & if \quad 'exclusive-or'\ group(g) \\ \left(\prod_{f \in g} \left(V(f)+1\right)\right) - 1 & if \quad 'inclusive-or'\ group(g) \end{cases}$$

For concepts, we calculate the features with *V(C) = V'(C)*, where we overload V'(.) to work on concepts.

Overview

- Basic Feature Modeling Concepts
- ➢ Exercise
- AmiEddie-XML 1.3
- Captain Feature
- Homework

Exercise

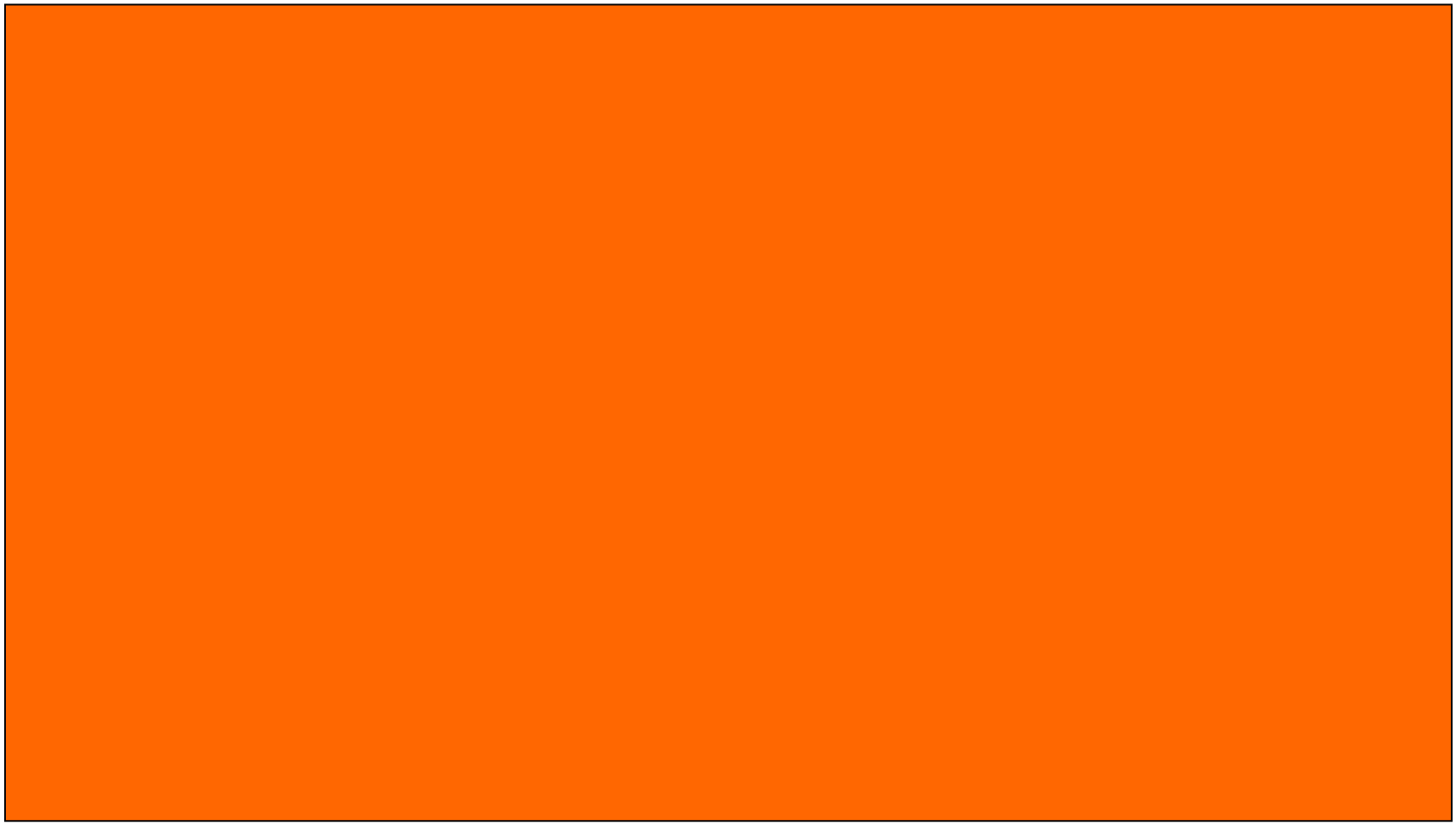
Draw a feature diagram for a family of counters

The following requirements hold:

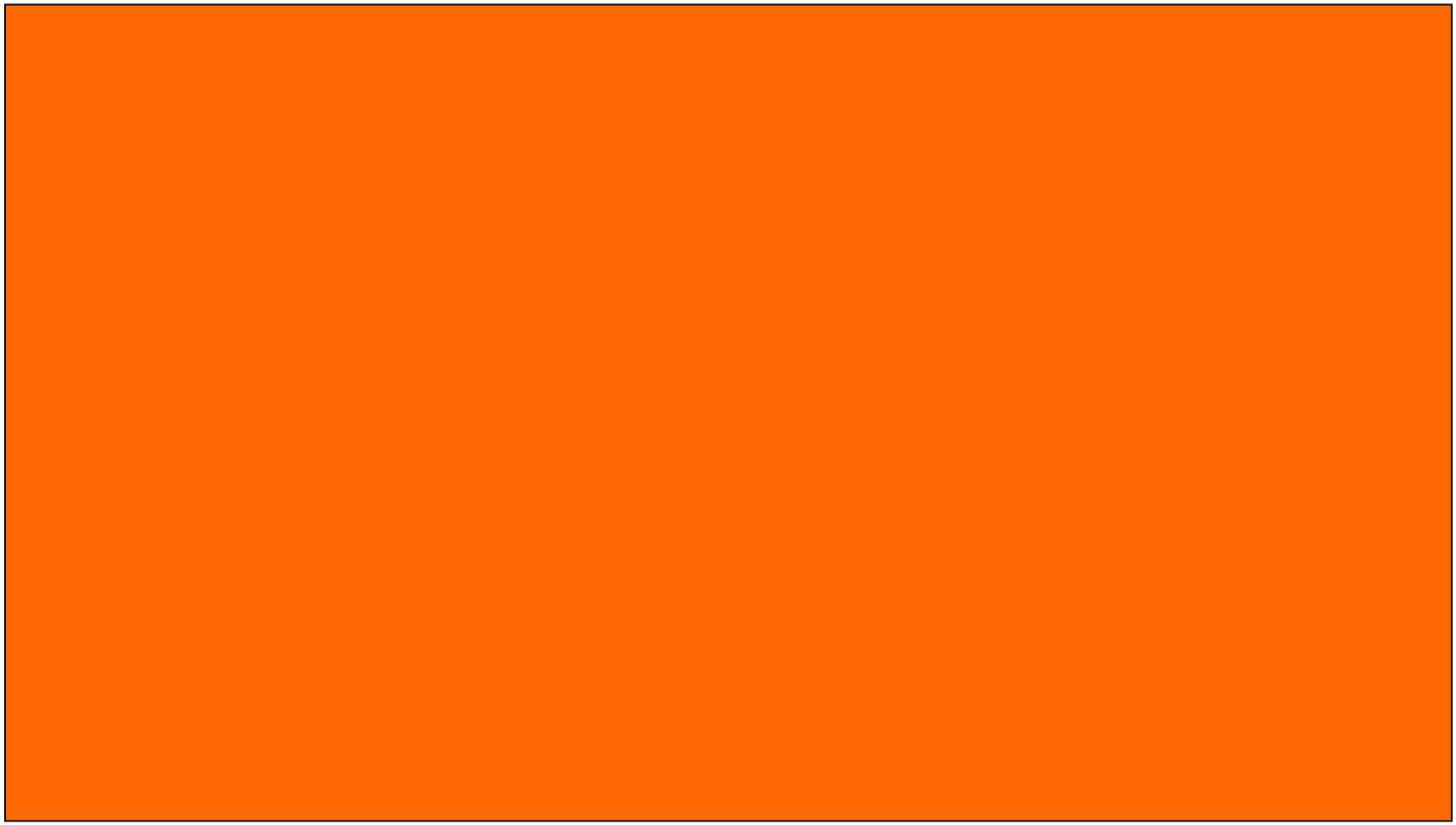
- support a fixed or variable increment
- the value of the fixed increment can be 'statically' specified
- support different counter value types (assume just integral, built-in types: char, short, int, long, unsigned char, unsigned short, unsigned int, unsigned long)
- assume that more value types may be added
- may optionally support manual or automatic reset or both; automatic reset is activated when the counter value exceeds a reset limit
- the reset limit can be 'statically' specified

How many different systems does this feature diagram describe?

Exercise: Feature Diagram



Exercise: Combinations



Overview

- Basic Feature Modeling Concepts
- Exercise
- ➢ AmiEddi-XML 1.3
- Captain Feature
- Homework

AmiEddi-XML 1.3

- GUI-based tool for interactively editing feature models
- Feature models and feature diagrams are part of a repository
  - The repository is stored in XML format
- Each feature model is associated with a meta-model for additional information
  - The meta-model can be edited
  - This way it is possible to capture additional information
  - Some additional information can also be visualized in a feature diagram

AmiEddi-XML 1.3

- Developers
  - Mario Selbig, feature model editor
  - Frank Blinn, configurable meta-model
  - Markus Lang, XML repository
- Free download from

  http://www.generative-programming.org

Exercise in AmiEddi

...

Overview

- Elements and Notation
- Exercise
- AmiEddi-XML 1.3
- ➢ Captain Feature
- Homework

New Requirements ...

... emerged from applying feature modeling and AmiEddi in real-world projects

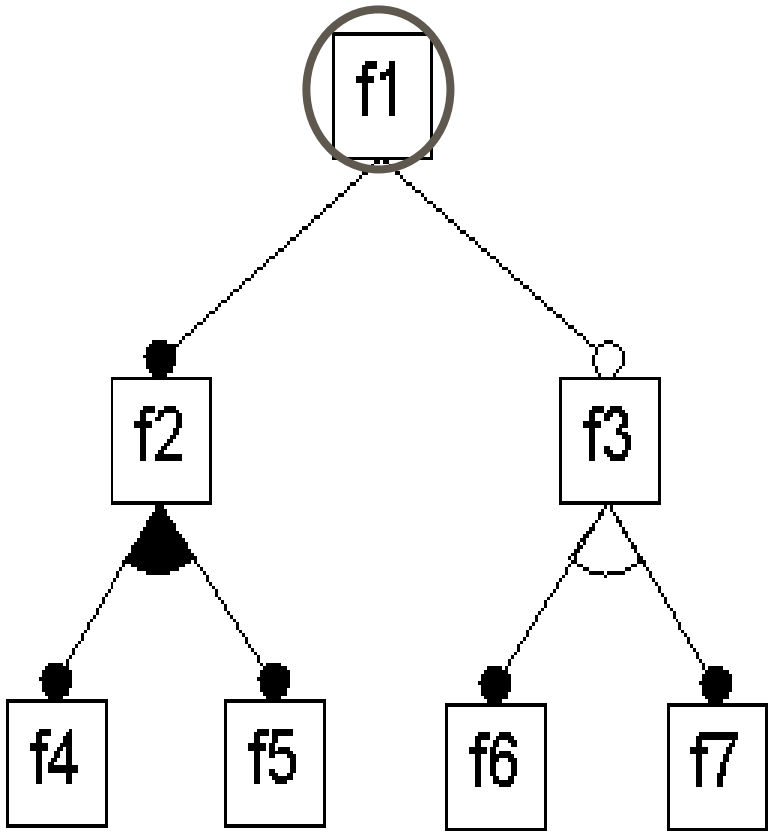
- Modularization of complex feature diagrams
- Flexible cardinalities of features and groups
- Extending the meta-model by the user
- Specialization of feature diagrams and creating system configurations
- Assigning values to features during specialization

➔Integrating feature modeling, meta-modeling, and specialization of feature diagrams as well as creating system configurations

Captain Feature

- Captain Feature is the successor of AmiEddi-XML and meets the afore-mentioned requirements
- Joint work with DaimlerChrysler Research Ulm
- Development
  - Thomas Bednasch
  - Markus Lang
  - Henrik Becker
  - James Markle
- Captain Feature is an open source project
  - *https://sourceforge.net/projects/captainfeature/*

Elements of Feature Diagrams



- Tree structure
- Exactly one concept node
- Mandatory feature
- Optional feature
- Inclusive-or group
- Exclusive-or group

Technical Concepts

- Feature
  - Comprises all properties of a feature
  - Is never used in diagrams and thus not visible in diagrams
  - Can be referenced in an arbitrary number of diagrams
- Feature node
  - The "visible" representation of a feature in a diagram
  - References a feature or - because of modularization - a concept node (representative)
  - During specialization a value can be assigned to it

Technical Concepts

- Cardinality
  - Denotes, how often a representative may appear in a configuration
  - AmiEddi allows 0-1 and 1-1 (optional and mandatory feature)
  - Captain Feature has flexible cardinalities, e.g. 2-4
- Group cardinality
  - Denotes, how many variants of a group may appear in a configuration
  - AmiEddi allows exclusive-or (1-1) and inclusive-or groups only (1-n)
  - Captain Feature has flexible groups, e.g. 3-5

Concepts in EBNF

<Feature> := Name Description [Premature] [<Type>]

<Type> := (Bool | Int | Float | String |Pointer)

<FeatureNode> := Pointer [<Cardinality> | <Group>]*

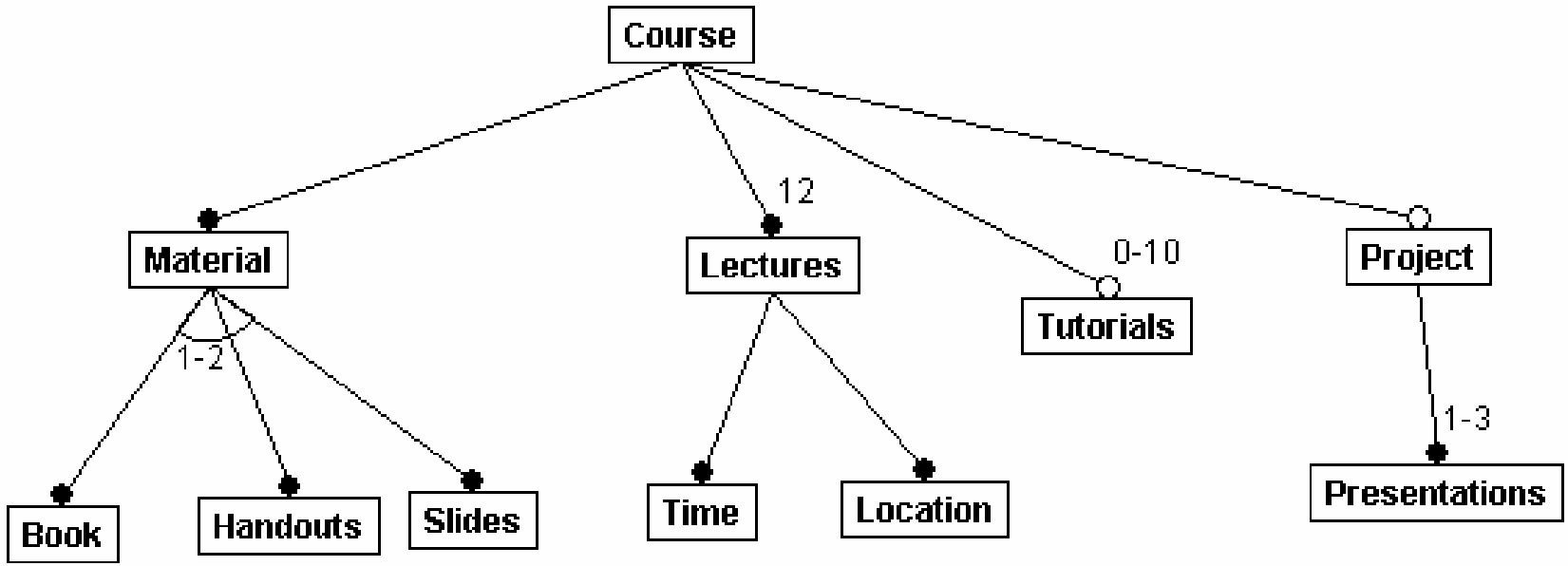
<Cardinality> := Minimum [Maximum] <FeatureNode> *)

<Group> := Minimum [Maximum] <Cardinality> <Cardinality>+

<Pointer> := FeaturePointer | FeatureNodePointer

*) Expect some changes of the cardinality concept

Example

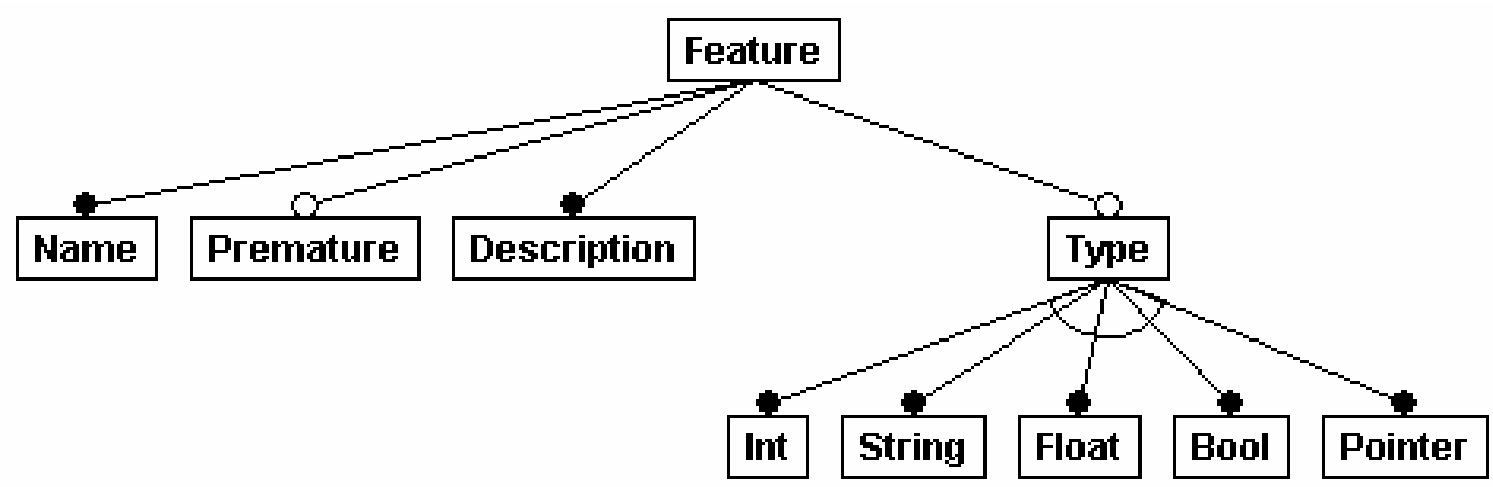


Meta-Model as Feature Model

- Idea
  - Use feature diagrams to describe the technical concepts
  - These so called "system diagrams" are the core of the meta-model
- Consequence
  - The designer can use the feature model editor to enhance the meta-model
  - This turns feature modeling into a complete, self-descriptive system

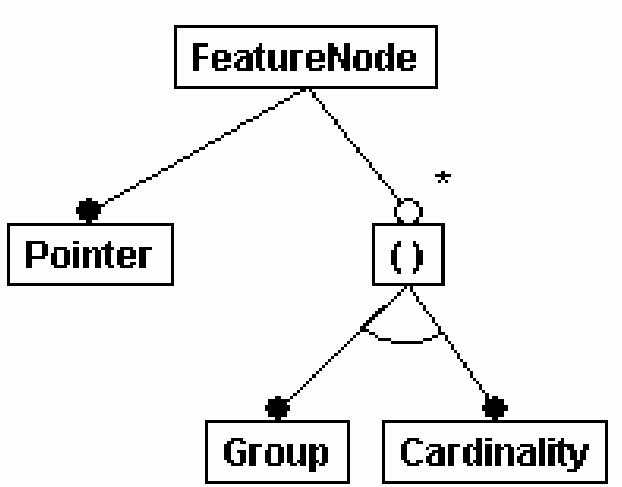
System Diagram "Feature"

<Feature> := Name Description [Premature] [<Type>]



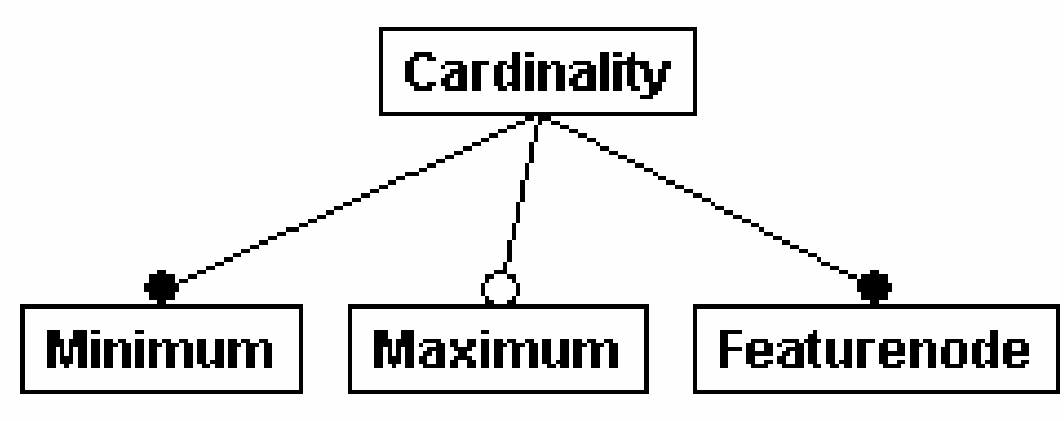
System Diagram "FeatureNode"

<FeatureNode> := Pointer [<Cardinality> | <Group>]*



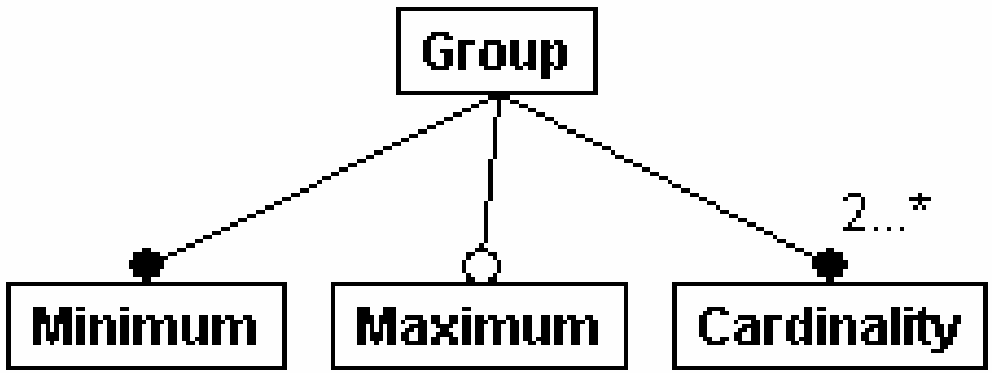
System Diagram "Cardinality"

<Cardinality> := Minimum [Maximum] <FeatureNode>

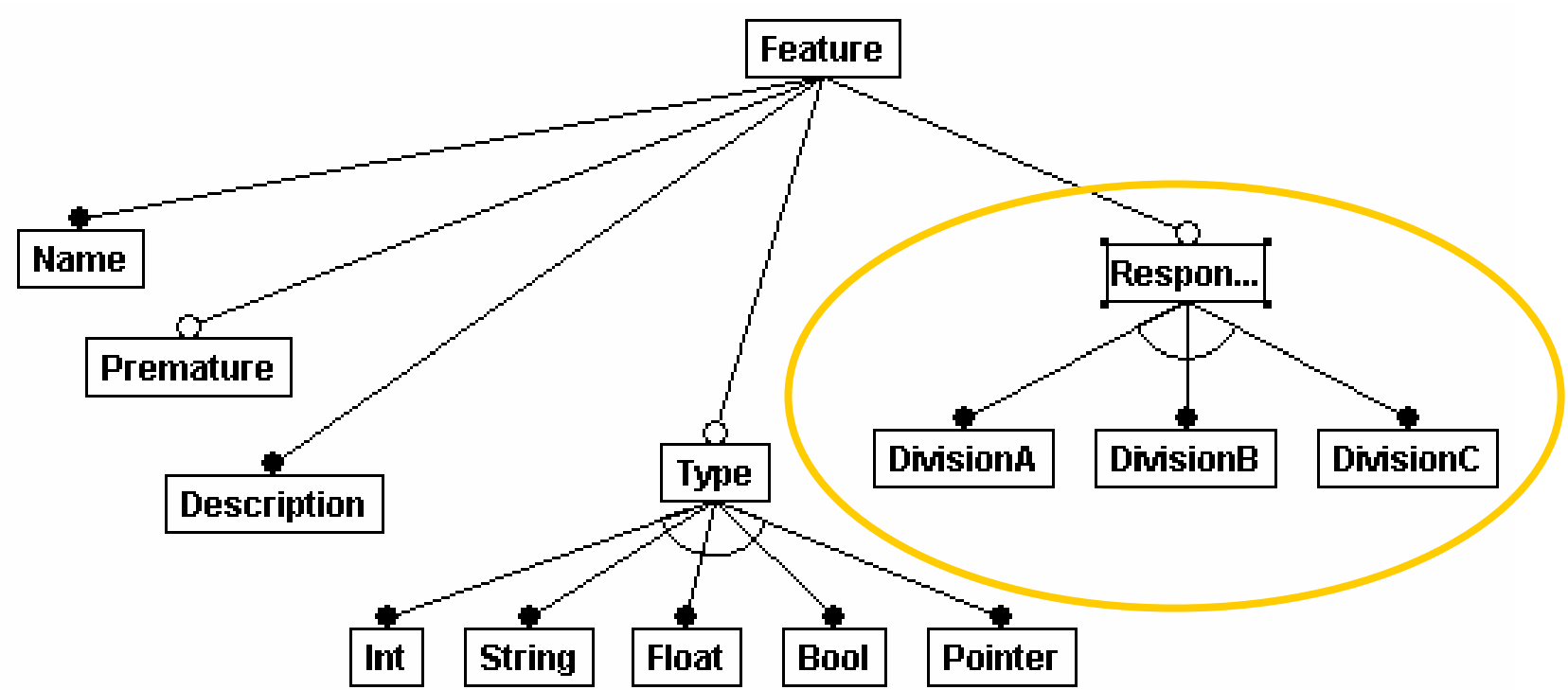


System Diagram "Group"

<Group> := Minimum [Maximum] <Cardinality> <Cardinality>+



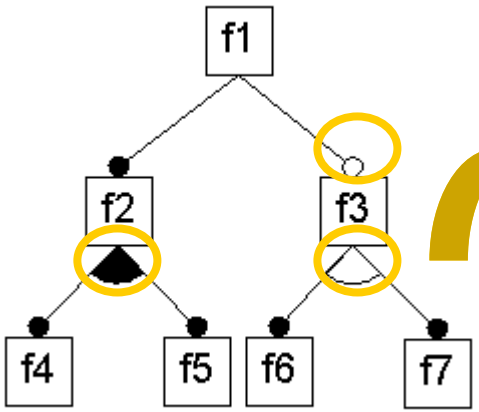
Extending the Meta-Model (Example)



Specialization and Configuration

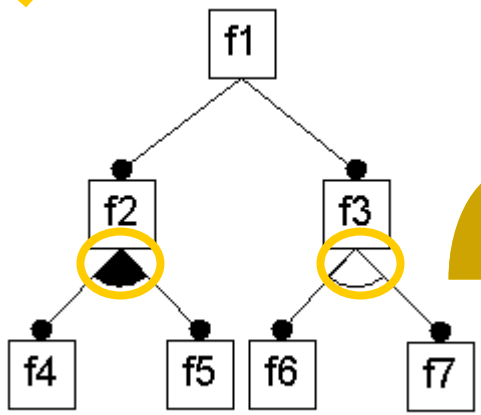
- Specialization means to remove variability of a feature diagram
  - Breaking groups
  - Denoting the exact cardinality
- The (partial) specialization of a feature diagram produces a new feature diagram
- Any feature-diagram may be declared as configuration, even if it stills contains variability
  - Either the generator decides itself whether to resolve remaining variability during generation or to include the variability in the generated systems
  - or the binding time has to be provided explicitly as additional information

Specialization (Example)



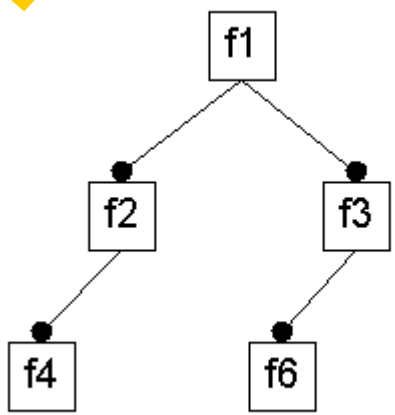
Partial specialization

Variability
- f3 is optional
- Inclusive-or group f4/f5
- Exclusive-or group f6/f7



Full specialization

Variability
- Inclusive-or group f4/f5
- Exclusive-or group f6/f7

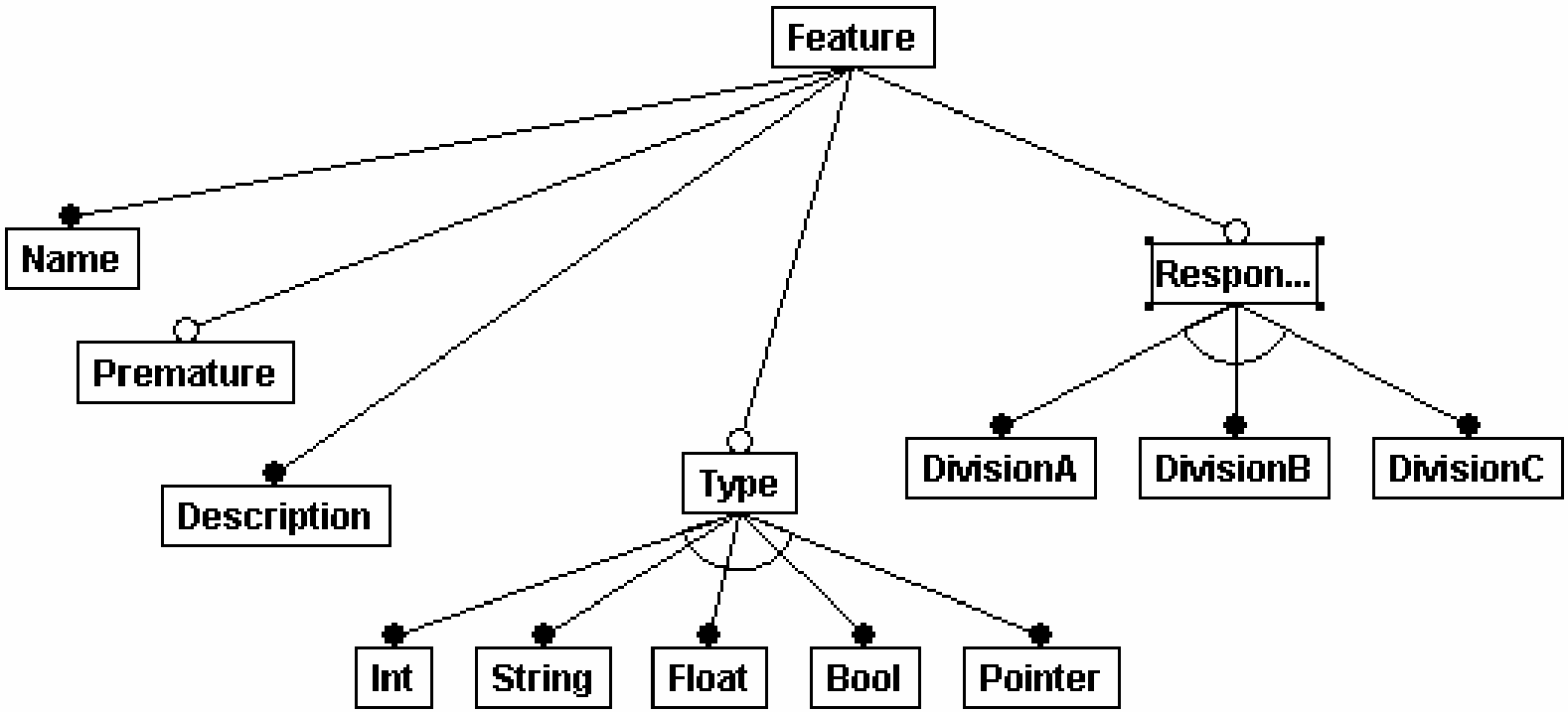


Specializing the Meta-Model

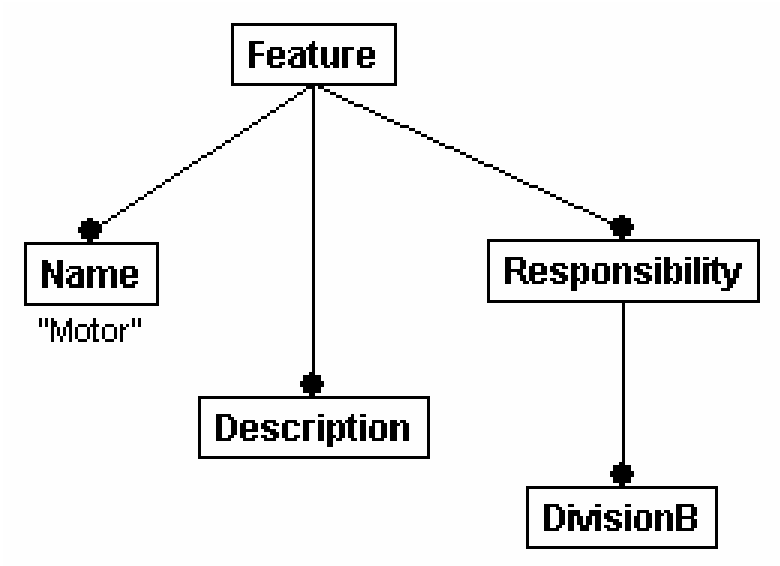
- The meta-model is also rendered as a feature model, thus specialization can be applied to the meta-model as well
- The specialization of system diagrams creates the elements a feature diagram consists of

Specializing the Meta-Model (Example)

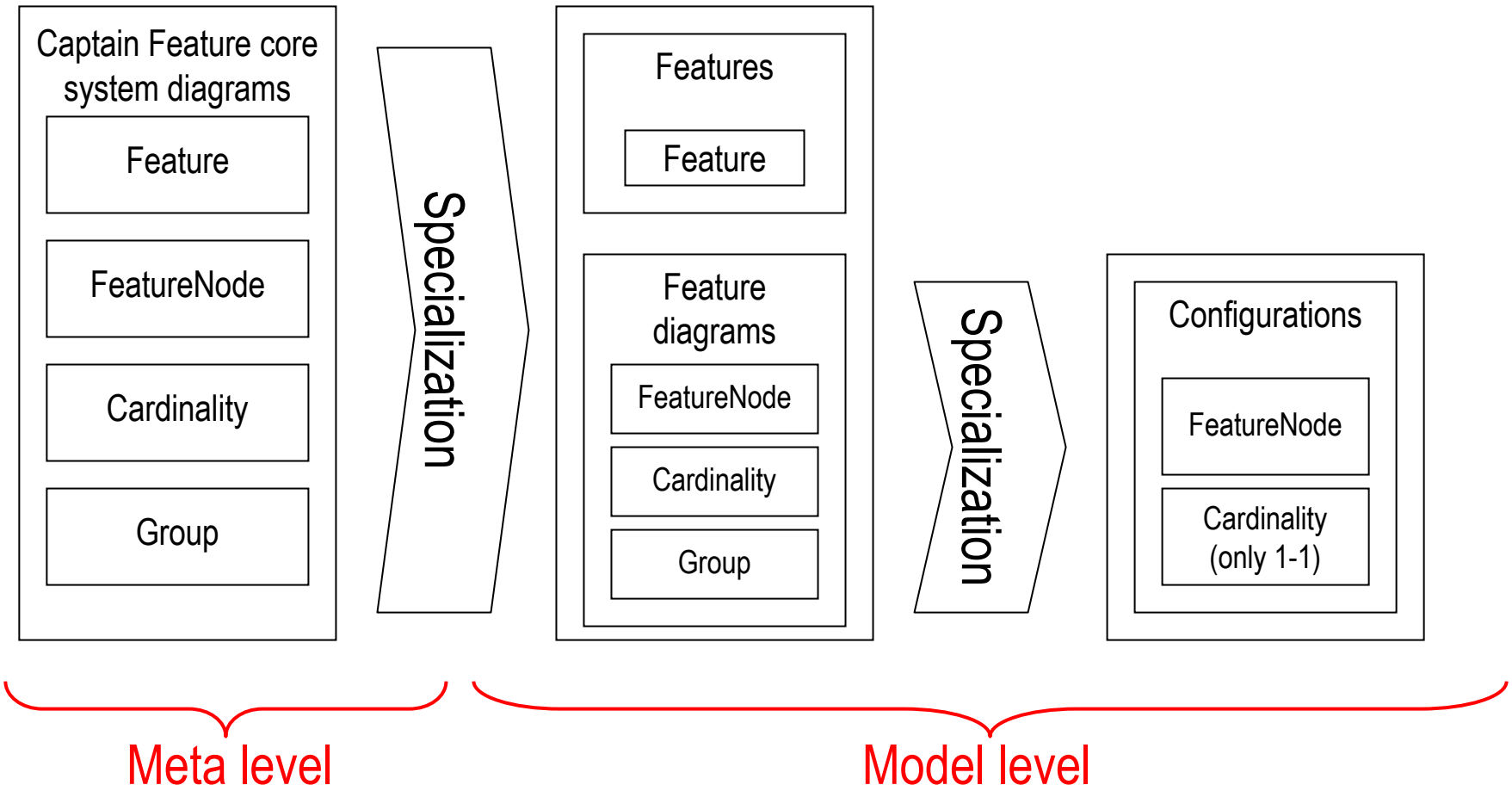
Specializing this system diagram could produce a feature named “Motor", for which "DivisionB" ist responsible (next slide)



Specializing the Meta-Model (Example)



Specialization (Overview)



Specializing in Captain Feature

...

Overview

- Basic Feature Modeling Concepts
- Exercise
- AmiEddi-XML 1.3
- Captain Feature
- ➢ Homework

Homework

1. Pick an application domain of your choosing (e.g. a mobile phone, a particular type of website, etc.)
2. Determine relevant common and variable features to describe a domain family
3. Produce a feature diagram and indicate any potential additional (meta-)information you might want
4. Next lecture, one or two people will be asked to present their feature model

Relevant chapters of the book: 2 and 4