O

O S E

E

Feature Modeling

Operating-System Engineering

What is the Use of That?

Feature models produced during featuring modeling provide us with an abstract (because it is implementation independent), concise, and explicit representation of the variability present in the software. And you'll find feature modeling to be useful at any level: You can use it at the system level, at the subsystem level, as well as down to classes and procedures. ([1], p. 82)

Features Revisited

- features allow us to express the commonalities and differences
  - organized in diagrams, they express the configurability aspect of concepts
- features are primarily used in order to discriminate between instances
  - the quality of a feature is related to properties
    - such as its primitiveness, generality, and independency
  - a feature aims at making distinctions between choices
- features represent reusable, configurable requirements

Feature Modeling and Feature Model

Feature modeling is the activity of modeling the common and the variable properties of concepts and their interdependencies and organizing them into a coherent model referred to as a feature model.
By concept, we mean any elements and structures in the domain of interest. ([1], p. 83)

Feature Model

- represents the common and the variable features of concept instances
  - including the dependencies between the variable features
- consists of a <u>feature diagram</u> and some additional information:
  - short semantic descriptions and/or rationales of each feature
  - stakeholders and client programs interested in each feature
  - examples of systems with a given feature
  - constraints, default dependency rules, availability and/or binding sites
  - binding models, open/closed attributes, and priorities
- represents the *intention* of a concept

Feature Diagrams

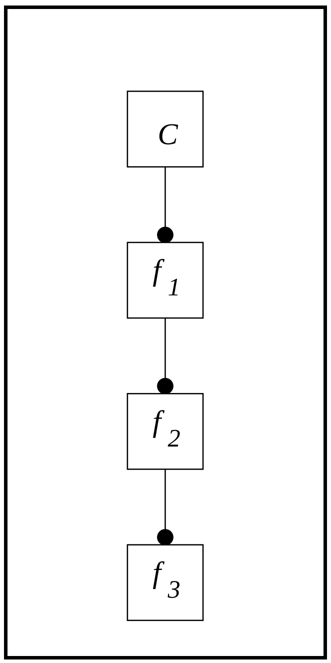
- consist of a set of nodes, directed edges, and edge decorations
  - edge decorations are drawn as arcs
    - connecting subsets or all of the edges originating from the same node
  - they define a partitioning of the subnodes of a node

- distinguish between two types of nodes:

concept node the root of a feature diagram, in short “concept” C
feature node any other node of a feature diagram, in short “feature” f_i

- the parent node of a feature is either the concept or another feature

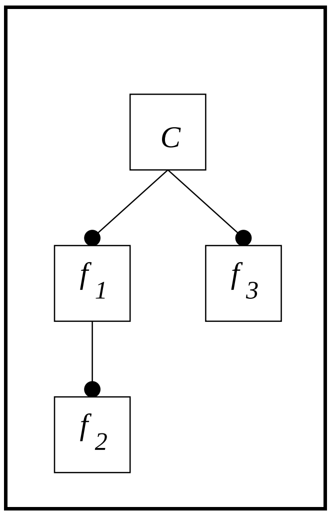
Notions of Feature and Feature Diagram



- given is a feature diagram with three features f_1, f_2, and f_3 of the concept C, where the relationships are specified as on the left, then . . .
  - f_1 is a *direct feature* of C
  - f_2 and f_3 are *indirect subfeatures* of C
  - f_2 is a *direct subfeatures* of f_1
  - f_3 is an *indirect subfeatures* of f_1
- according to FODA [3], there are several feature categories

Feature Categories

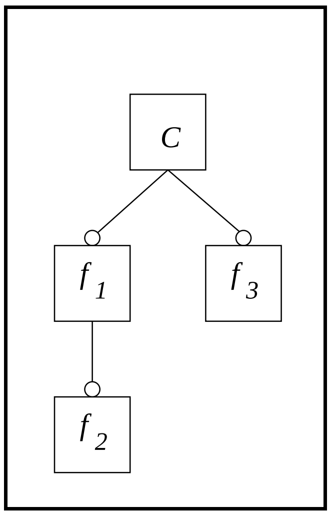
Mandatory Features



- a *mandatory feature* is included in the description of a concept instance if and only if its parent is included in the description of the instance
- if the parent of a mandatory feature is optional and not included in the instance description, the mandatory feature cannot be part of the description

- every instance of C can be described by the *feature set* $\{C, f_1, f_2, f_3\}$

Optional Features

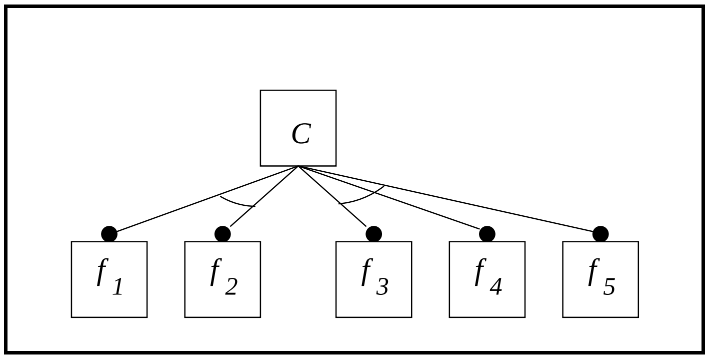


- an *optional feature* may be included in the description of a concept instance if and only if its parent is included in the description of the instance

- if the parent is included, the optional feature may be part of the description or not; if the parent is not included, the optional feature cannot be part of

- every instance of C can be described by the feature set $\{\{C\}, \{C, f_1\}, \{C, f_3\}, \{C, f_1, f_2\}, \{C, f_1, f_3\}, \{C, f_1, f_2, f_3\}\}$

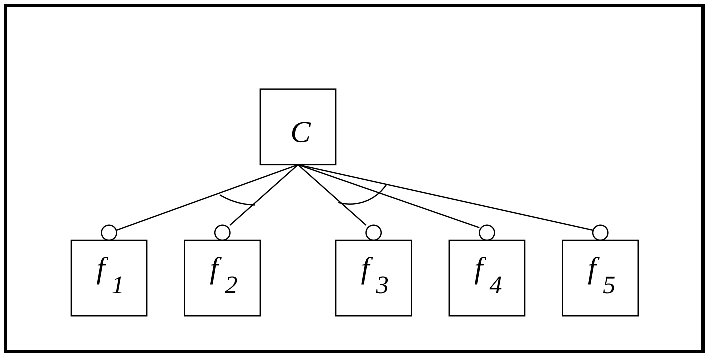
Alternative Features



- a concept may have one or more sets of direct *alternative features*
- a feature may have one or more sets of direct *alternative subfeatures*

- if the parent of a set of alternative features is included in the description of a concept instance, then exactly one feature from this set of alternative features is included in the description; otherwise none are included
- every instance of C can be described by the feature set $\{\{C, f_1, f_3\}, \{C, f_1, f_4\}, \{C, f_1, f_5\}, \{C, f_2, f_3\}, \{C, f_2, f_4\}, \{C, f_2, f_5\}\}$

Optional Alternative Features

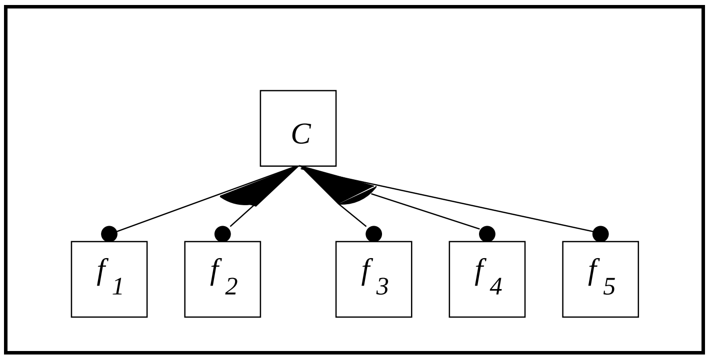


- a concept may have one or more sets of direct *optional alternative features*
- a feature may have one or more sets of direct *optional alternative subfeatures*

- if the parent of a set of optional alternative features is included in the description of a concept instance, then exactly one feature from this set may be included in the description; otherwise none are included

- every instance of C can be described by the feature set $\{\{C\}, \{C, f_1\}, \{C, f_2\}, \{C, f_3\}, \{C, f_4\}, \{C, f_5\}, \{C, f_1, f_3\}, \{C, f_1, f_4\}, \{C, f_1, f_5\}, \{C, f_2, f_3\}, \{C, f_2, f_4\}, \{C, f_2, f_5\}\}$

Or-Features



- a concept may have one or more sets of direct *or-features*
- a feature may have one or more sets of direct *alternative or-subfeatures*
- if the parent of a set of or-features is included in the description of a concept instance, then any nonempty subset from the set of or-features is included in the description; otherwise none are included
- a total of $(2^2 - 1) \times (2^3 - 1)$ different descriptions of instances of C can be derived from this diagram, i.e. the feature set consists of 21 elements

Normalized Feature Diagrams

- the combination of the basic feature categories ($\rightarrow$ p. 7) may result in:

  optional alternative features [1] If one or more of the features in a set of alternative features is optional, it has the same effect as if all the alternative features in this set were optional.

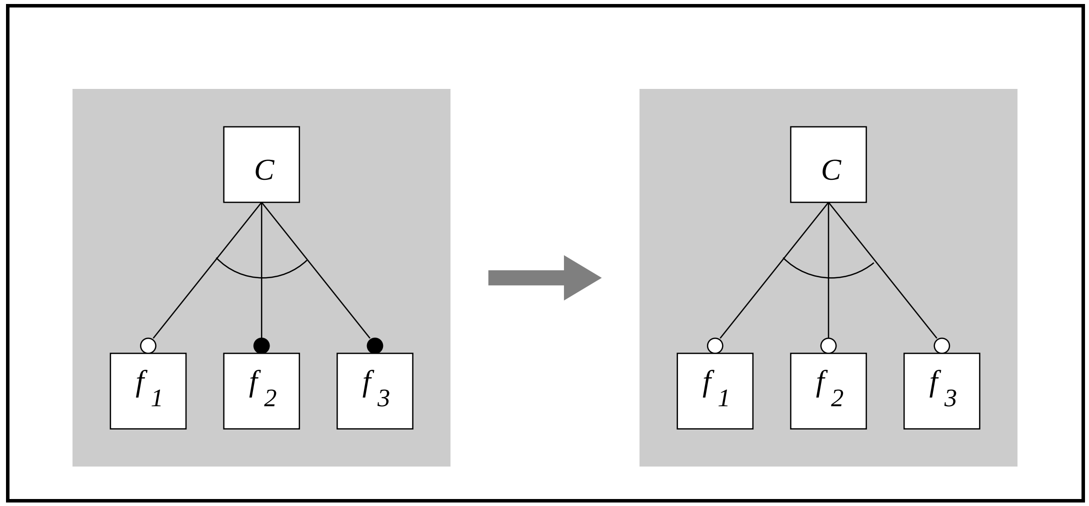
  optional or-features If one or more of the features in a set of or-features is optional, it has the same effect as if all the features in this set were optional.

- the category of optional or-features is redundant

[1] In difference to the category of the *optional alternative feature* shown before ($\rightarrow$ p. 11), the category discussed here consists of a combination of optional <u>and</u> alternative features in the same set. Thus, the set is inhomogeneous, i.e., not constituted by elements of the same feature type.

Equivalence of Feature Diagrams — Optional Alt. Features

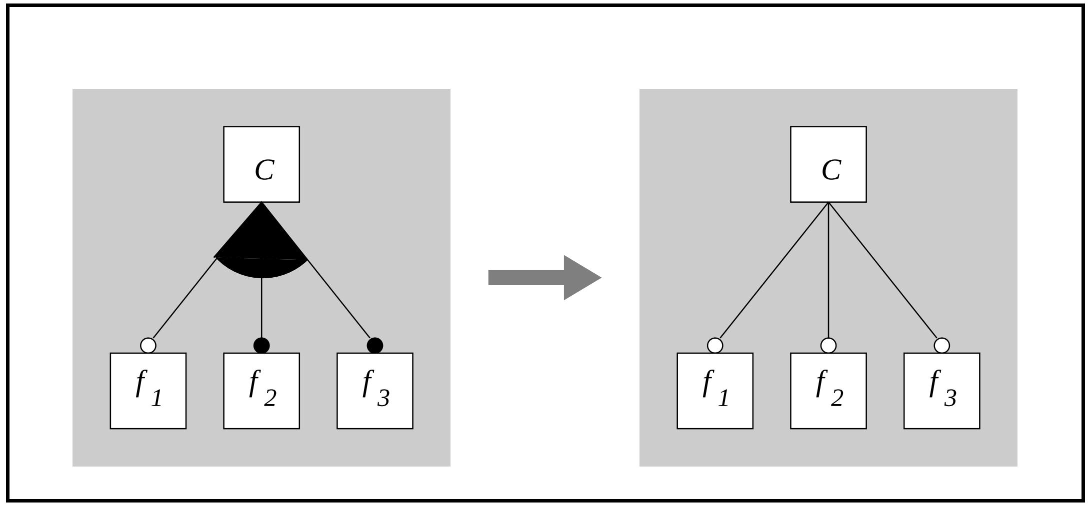
The presence of the optional alternative feature f_1 allows one to derive the empty subset from the set of all subfeatures of C. On the other hand, at most exactly one feature may be included in the description of an instance of C. This means that all features of C are optional alternative.



- a feature diagram with one (or more) optional alternative feature and one (or more) alternative feature is equivalent to a feature diagram with all optional alternative features

Equivalence of Feature Diagrams — Optional Or-Features

The presence of the optional or-feature f_1 allows one to derive the empty subset from the set of all or-features of C. Thus, the feature set consists of any combination of features, including the empty set. This means that all features of C are optional.



- a feature diagram with one (or more) optional or-feature and one (or more) or-feature is equivalent to a feature diagram with all optional features

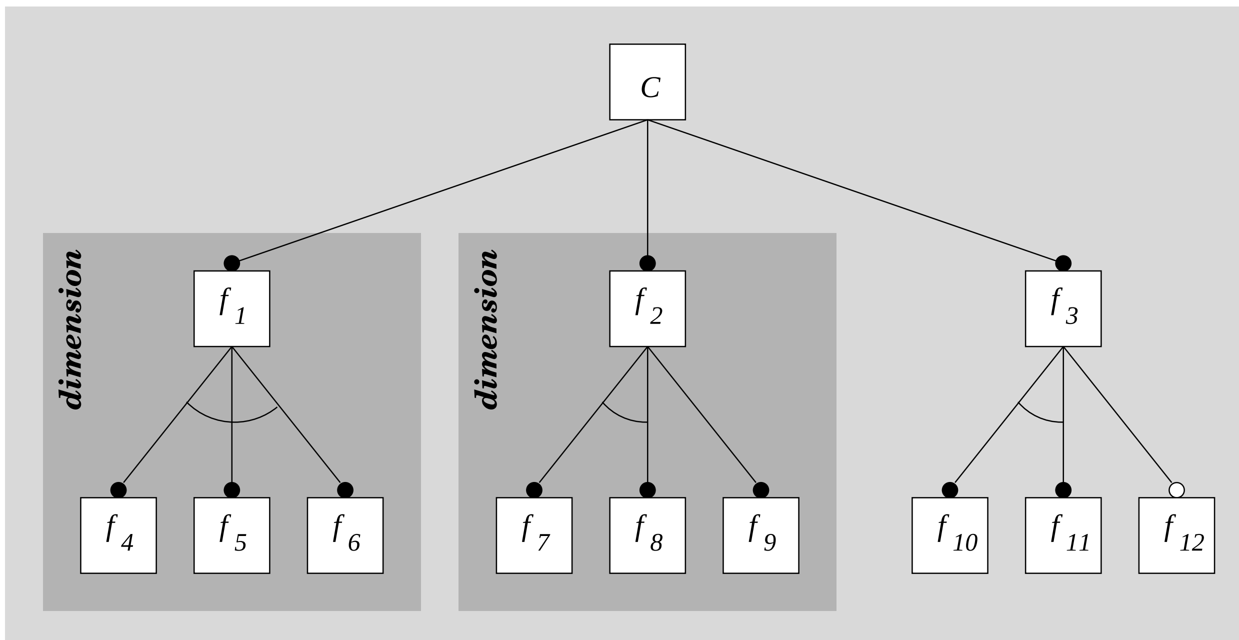
Feature Diagrams and Dimensions

A feature (or concept) with a single set of direct alternative subfeatures (or features) and no other direct subfeatures (or features) is referred to as a *dimension*.

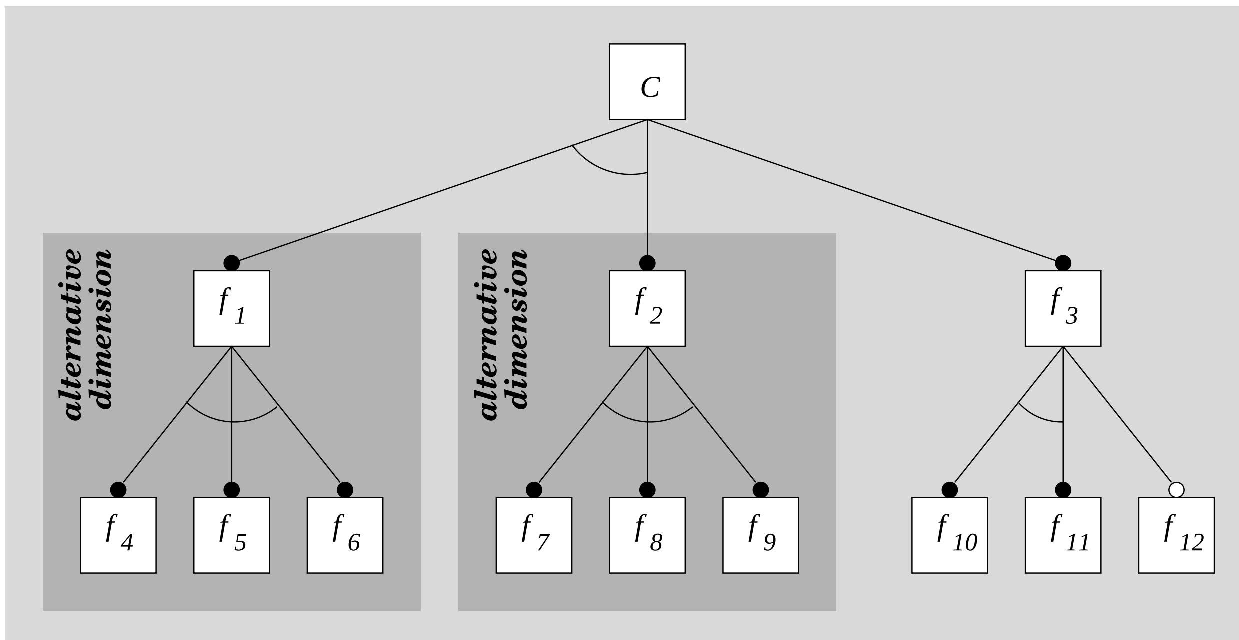
We also found it useful to broaden the notion of dimensions to include features (concepts) with a single set of direct alternative subfeatures (features) and one or more direct mandatory subfeatures (features).

Dimensions can also be alternative, that is, we can have *alternative dimensions*. ([1], p. 90)

Dimensions



Alternative Dimensions



Expressing Commonality in Feature Diagrams

common feature is a feature present in all instances of a concept:

- all mandatory features whose parent is the concept are common features
- all mandatory features whose parents are common are themselves common

common subfeature of f_i is a (direct or indirect) subfeature of f_i, which is present in all instances of a concept that also have f_i:

- all direct mandatory subfeatures of f_i are common subfeatures of f_i
- a subfeature of f_i is common if it is mandatory and there is a path of mandatory features connecting the subfeature and f_i

Expressing Variability in Feature Diagrams

- variability in feature diagrams is expresed using *variable features*
  - i.e., the optional, alternative, optional alternative, and or-features
- nodes at which these features are attached are referred to as *variation points*
  - features (or concepts) with at least one direct variable subfeature (or feature)
- distinguished between *homogeneous* versus *inhomogeneous variation points*

  A variation point is homogeneous if all its direct subfeatures (or features) belong to the same subnode category (i.e., they are all optional, or all alternative, and so on); otherwise it is inhomogeneous. ([1], p. 96)

More on Variation Points

singular vs. nonsingular variation points Variation points allowing one to include at most one direct variable subfeature (or feature) in the description of a concept are attributed *singular*; they are *nonsingular* in case of more than one direct variable subfeature (or feature).

mutually exclusive features Two features in a feature diagram of a concept are mutually exclusive if none of the instances of the concept has both features at the same time.

simultaneous vs. nonsimultaneous variation points Two variation points of a feature diagram are *simultaneous* if and only if they are not mutually exclusive and neither of them is a direct or indirect variable subfeature of each other; otherwise they are *nonsimultaneous*.

How to Find Features

- strategies for identifying features may be both bottom-up and top-down:
  - look for important domain terminology that implies variability
  - examine domain concepts for different sources of variability
  - use *feature starter sets* [2] to start the analysis
  - look for features at any point in the development
  - identify more features than you initially intend to implement
- finding of features is a creative process, assumes some amount of “experience”

[2] A feature starter set consists of a set of perspectives for modeling concepts. Some combinations of perspectives are more appropriate for a given domain than for another. Starter sets can be understood as reusable resources capturing modeling experience.

Feature Modeling Process

- feature modeling is a continuous, iterative process with the following steps:
  1. record similarities between instances, i.e., common features
  2. record differences between instances, i.e., variable features
  3. organize features in feature diagrams, i.e., into hierarchies
  4. analyze feature combinations and interactions
  5. record all the additional information regarding features
- these steps make up the *micro-cycle* of feature modeling
  - because they are usually executed in small, quick cycles
- the quality of the result depends on the modeler's skills or domain knowledge

Concluding Remarks

Feature modeling is the greatest contribution of Domain Engineering to software engineering. Feature modeling is a must if you engineer for reuse. This is because reusable software contains inherently more variability than concrete applications and feature modeling is the key technique for identifying and capturing variability. ([1], p. 82)

Bibliography

[1] K. Czarnecki and U. W. Eisenecker. *Generative Programming—Methods, Tools, and Applications*. Addison-Wesley, 2000. ISBN 0-201-30977-7.

[2] E. W. Dijkstra. *A Principle of Programming*. Prentice Hall, Englewood Cliffs, NJ, 1976.

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