Feature Modeling
What is the use of that?
features represent reusable, configurable requirements

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

Features Revisited
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 intention of a concept

- binding models, open/closed attributes, and priorities
- constraints, default dependency rules, availability and/or binding sites
- examples of systems with a given feature
- stakeholders and client programs interested in each feature
- short semantic descriptions and/or rationales of each feature
- binding models, open/closed attributes, and priorities
Feature diagrams consist of a set of nodes, directed edges, and edge decorations. Directed edges are drawn as arcs and edge decorations are drawn near edges. They define a partitioning of the subnodes of a node. The root of a feature diagram, in short, “feature,” distinguishes between two types of nodes:

- Concept node is the root of a feature diagram, in short, “concept,” and its parent node of a feature is either the concept or another feature.
- Feature node, any other node of a feature diagram, in short, “feature,” distinguishes subsets of all the edges originating from the same node. Edges and edge decorations are drawn near edges, and edge decorations consist of a set of nodes, directed edges, and edge decorations.

Feature Diagrams
Notions of Feature and Feature Diagram

Given is a feature diagram with three features $f_1$, $f_2$, and $f_3$ according to FODA [3], there are several feature categories:

- $f_3$ is an indirect subfeature of $f_1$ if $f_2$ is a direct subfeature of $f_1$.
- $f_2$ and $f_3$ are indirect subfeatures of $C$ if $f_1$ is a direct feature of $C$.

On the left, the relationships are specified as:

Operating-System Engineering Feature Modeling
Feature Categories

- or-features
- optional alternative features
- alternative features
- optional features
- mandatory features

Feature Categories
every instance of \( C \) can be described by the feature set \( \{ f_1, f_2, f_3 \} \) if and only if its parent is included in the instance description. The mandatory feature cannot be part of the instance description if the parent of a mandatory feature is optional and not included in the instance description. If the parent of a mandatory feature is included in the instance description, the mandatory feature cannot be part of the instance description.
Optional Features

\{\{f_3, f_2, f_1\}, \{f_1\}, \{f_3\}\} 
\{\{f_3\}, \{f_1\}, \{f_1\}\} 
\{\{f_3\}, \{f_1\}\} 
\{\}\n
Every instance of \( C \) can be described by the feature set \( \{f_1, f_2, f_3\} \). If the parent is included, the optional feature may be part of the description of the instance. If the parent is not included, the optional feature cannot be part of the instance.

Optional Features
A concept may have one or more sets of alternative features. A feature may have one or more sets of alternative subfeatures. Every instance of $C$ can be described by the feature set $C', f_1, f_2, f_3, f_4, f_5, f_6$. 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 of an instance of $C$; otherwise none are included.

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.

Every instance of a concept can be described by the feature set 

\{ \{ f_5, f_4, f_5 \}, \{ f_2, f_3, f_2 \}, \{ f_1, f_1 \}, \{ f_4, f_3, f_1 \}, \{ f_1, f_3 \} \}

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.

Optional Alternative 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-sub-features. If the parent of a set of or-features is included in the description of a concept, then any nonempty subset from the set of or-features is included in the description of an instance, otherwise none are included.

A total of \((\frac{2^n - 1}{2 - 1}) \times (\frac{n}{2} - 1)\) different descriptions of instances of \(C\) can be derived from this diagram, i.e., the feature set consists of \(21\) elements.
The combination of the basic feature categories (p. 7) may result in:

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

- **alternative features**: 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 were optional.

In contrast to the category of the optional alternative feature shown before (p. 11), the category of optional or-features is redundant.
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.
The presence of the optional or-feature $f$ 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 or-features of $C$. This means that all features of $C$ are optional, including the empty set, or features. Therefore, a feature diagram with one (or more) optional or-feature and one (or more) optional or-feature is equivalent to a feature diagram with all optional features.
A feature (or concept) with a single set of direct alternatives (subfeatures) and no other direct mandatory subfeatures (features) is referred to as a **mandatory dimension**.

A feature (or concept) with a single set of direct alternative subfeatures (or features) is referred to as a **alternative 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) (1, p. 90).

**Feature Diagrams and Dimensions**
Dimensions
Alternative Dimensions
Expressing Commonality in Feature Diagrams

A feature common to all instances of a concept:
- All mandatory features whose parents are the concept are common.
- All mandatory features whose parents are common themselves are common.
- All direct mandatory subfeatures of a feature, which is a subfeature of, is a feature present in all instances of a concept that also have: common subfeature of.
- All mandatory features whose parents are common are the mandatory features present in all instances of a concept.
- All mandatory features whose parents are the concept are common features.
- All mandatory features common to all instances of a concept.

Operating-System Engineering — Feature Modeling
Expressing Variability in Feature Diagrams

Variability in feature diagrams is expressed 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. 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, or all optional alternative, and so on); otherwise, it is inhomogeneous ([1], p. 96).

Distinguished between homogeneous versus inhomogeneous variation points.

Operating-System Engineering | Feature Modeling
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) in the description of a concept.

Two variation points of a feature diagram are simultaneously exclusive if none of the instances of the concept has both features at the same time. Two features in a feature diagram of a concept are mutually exclusive if they are not simultaneously exclusive.

Twofeaturesin aan feature diagram ofaconceptare mutuallyexclusiveifnoneoftheinstancesoftheconcepthashothfeaturesatthesametime.Twovariation pointsofa feature diagram are simultaneously exclusive if and only if they are not mutually exclusive; otherwise, they are nonsimultaneous.}

singular vs. nonsingular variation points
Strategies for identifying features are both bottom-up and top-down:

Look for important domain interminology that implies variability
- Look for features at any point in the development
- Use feature starter sets to start the analysis
- Examine domain concepts for different sources of variability
- Look for important domain terminology that implies variability

Finding of features is a creative process, assumes some amount of “experience”

- Identify more features than you initially intend to implement
- Look for features anywhere in the development
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.

The quality of the result depends on the modeler's skills or domain knowledge — because they are usually executed in small, quick cycles, because they are usually executed in small, quick cycles.
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)
References


