Integrating Feature Modeling into UML

NODe 2006
Erfurt, Germany

Valentino Vranić and Ján Šnirc

Institute of Informatics and Software Engineering
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava

September 18, 2006

Overview

Feature Model

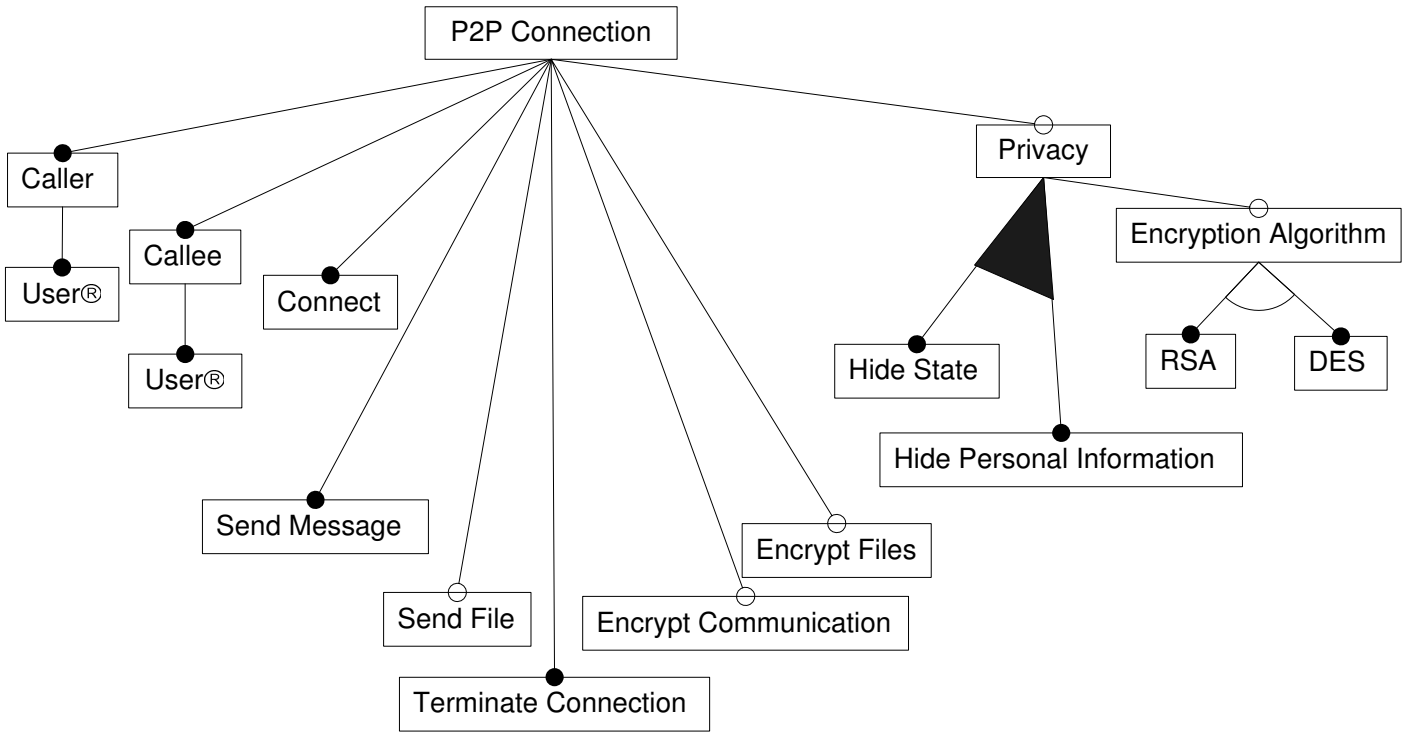
- Feature modeling enables to deal with variability abstractly
- Used in software product lines
- Concepts expressed by their features
- A feature is an important property of a concept
- Common and variable features
- Focus on configurability

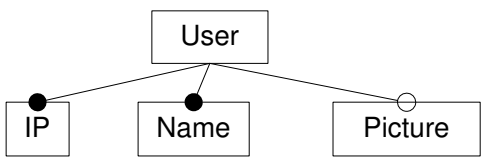
Feature Modeling and UML

- UML — de facto standard for modeling software
- Need to integrate feature modeling: to use it along with other UML models
- Feature modeling should be integrated into UML *correctly*

Basic Feature Modeling Notation

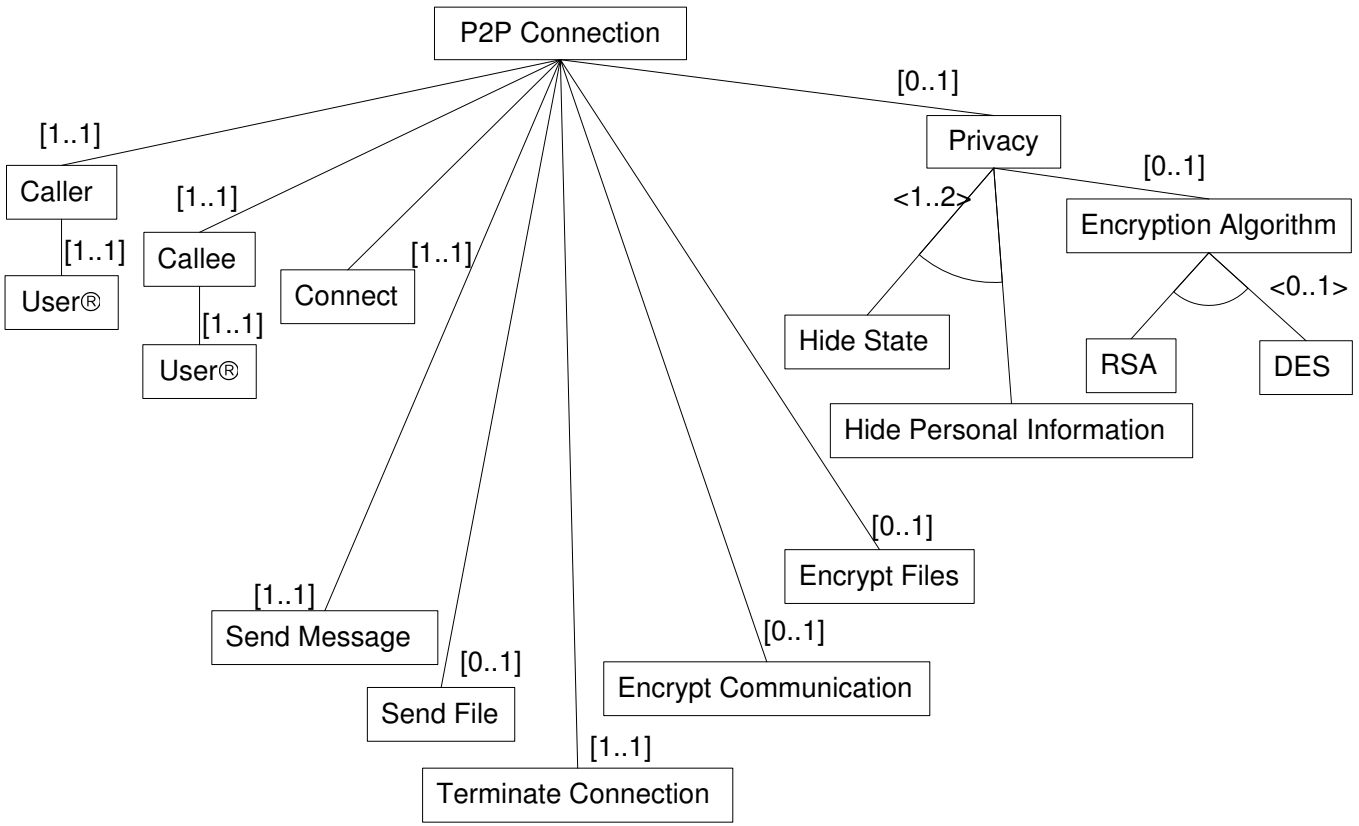
- Czarnecki-Eisenecker notation (based on FODA notation)





Feature Modeling with Cardinalities

- Cardinality-based Czarnecki-Eisenecker notation

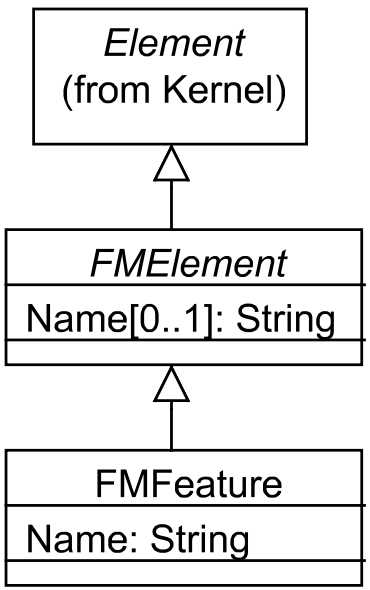


Keep Feature Modeling Elements Abstract in UML

- Concepts and features are abstract terms
- They should be represented abstractly in UML: without implementation connotation
- Some approaches use class stereotypes
- But classes can be in aggregation and inheritance relationships; features and concepts cannot

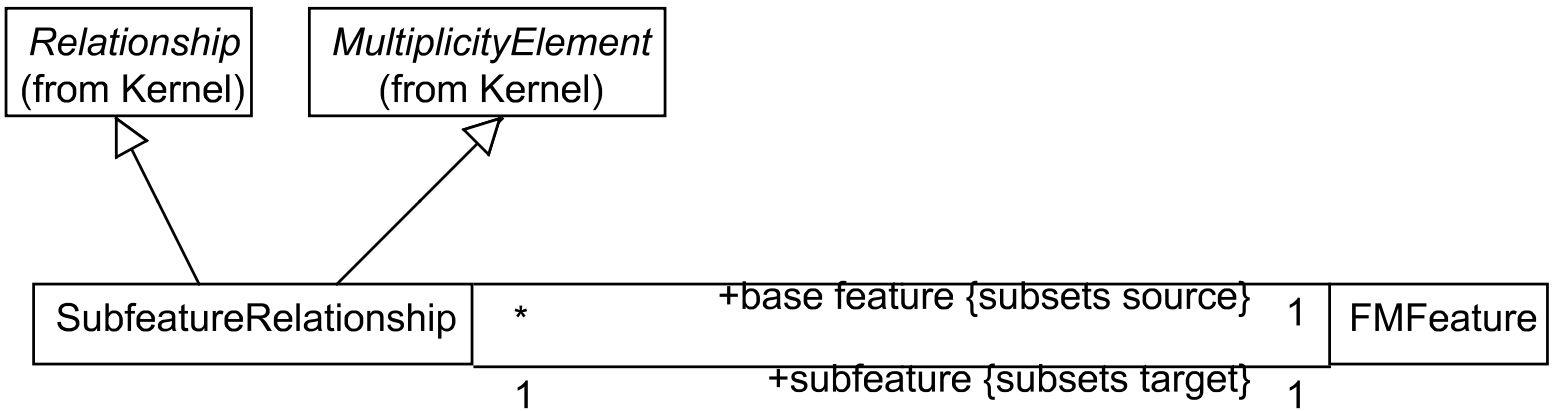
Concepts and Features

- Features and concepts are derived from **Element**, the most abstract UML metaclass for representing elements
- We introduce an abstract metaclass **FMElement** as a basis for all feature modeling elements other than relationships
- Concepts are considered to be "root features"



Subfeature Relationship

- Relationships between features do not have predefined semantics
- They are *not* aggregations, nor inheritance — this is decided later in design
- Therefore subfeature relationship is represented by a metaclass derived from the abstract **Relationship** metaclass

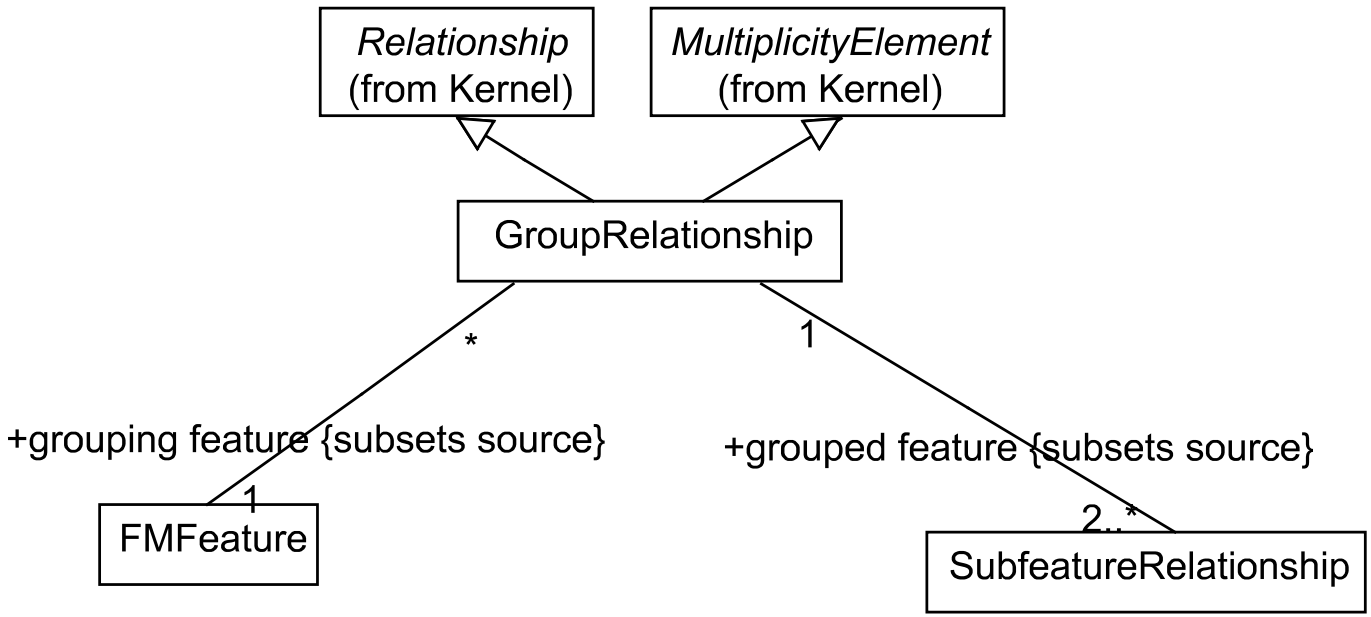


Subfeature Relationship (2)

- Cardinalities enabled by deriving **SubfeatureRelationship** from **MultiplicityElement**
- Variability information is a part of the relationship (edge), not feature (node)

Group Relationship

- A relationship between the base feature and two or more *subfeature relationships*, not features
- Based on an abstract **Relationship**



- Also based on **MultiplicityElement** to enable group cardinalities (how many features are to be included in a group)

Common Subfeature Types

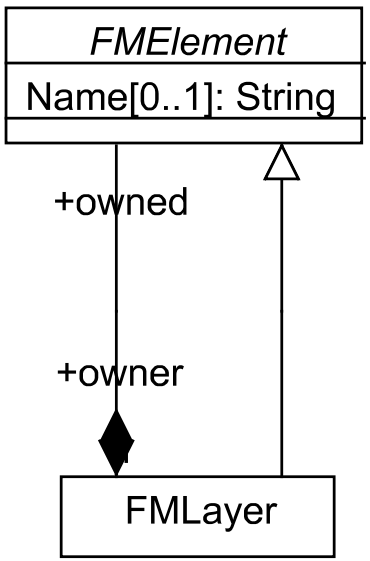
- Mandatory feature: **SubfeatureRelationship** with cardinality <1..1>
- Optional feature: **SubfeatureRelationship** with cardinality <0..1>
- Alternative feature: **GroupRelationship** with cardinality <1..1>
- Or-feature: **GroupRelationship** with cardinality <1..*>
- Common subfeature types can be defined as stereotypes along with their common graphical representation

Feature Diagrams as Graphs

- With this extension, feature diagrams may be modeled as graphs
- The tree representation is more common
- An **FMFeature** <<concept>> stereotype may be introduced
  - A feature that cannot be in a **subfeature** role in **SubfeatureRelationship**
  - "Root feature"

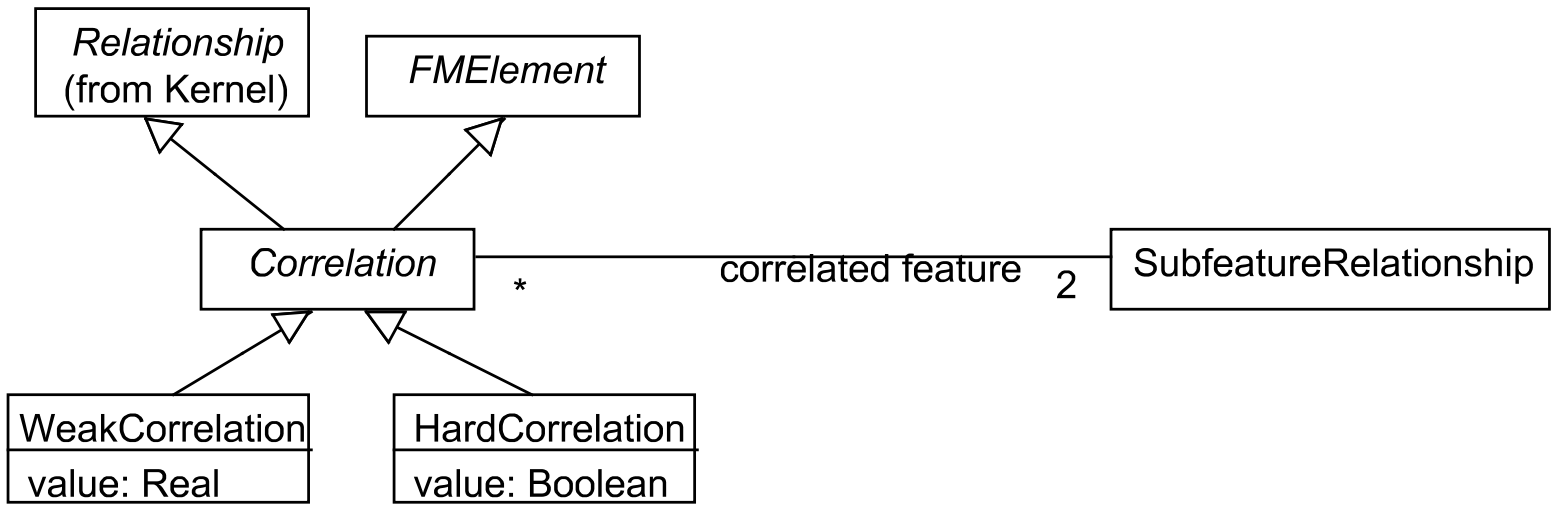
Layers

- Many notation specific extensions to feature modeling
- We consider two examples: layers and correlations
- Some approaches use layered representation of feature models (e.g., FODA and FORM)
- A layer is modeled as an **FMElement**



Correlation

- Design spaces
- Correlation as a form of expressing additional constraints between feature
  - Weak correlation: a recommendation to combining features
  - Hard correlation: combine or don't combine the features
- A design space is modeled as a **SubfeatureRelationship**



Feature Modeling Extension and UML Modeling Tools

- Not possible to modify the UML metamodel in contemporary UML modeling tools
- We also created a feature modeling profile for UML
- Enables experimentation with combining different feature modeling elements
- But the abstractness of feature modeling elements cannot be achieved this way
- However, feature variability is modeled as relationships

Conclusions

- An approach to integrate feature modeling into UML by extending its metamodel
- Feature modeling elements derived from basic UML elements to preserve their abstractness
- The extension provides a basis for further development
- The issue of notation-specific extensions has also been addressed

Further Work

- Introducing concept references
- Covering additional information on concepts and features (e.g., description and binding tiem information)
- Representing Constraints and default dependency rules as logical expressions using OCL
- Further experimentation with other notation-specific extensions
- Transformation of feature models in an MDA manner