



FAKULTÄT FÜR
INFORMATIK

How to Use FeatureIDE

Thomas Thüm

July 15, 2010

Content

- ▶ What is Feature-Oriented Software Development?
- ▶ What functionality does FeatureIDE provide?
- ▶ How to start working with FeatureIDE?



Content

- ▶ What is Feature-Oriented Software Development?
 - ▶ Feature-Oriented Programming + Example
 - ▶ Configurations
 - ▶ Feature Model
 - ▶ Composition Engines
- ▶ What functionality does FeatureIDE provide?
- ▶ How to start working with FeatureIDE?



Feature-Oriented Programming (FOP)

- ▶ Introduced 1997 by Christian Prehofer
- ▶ Based on Object-Oriented Programming
- ▶ Features realize functionalities
- ▶ Features are cross-cutting to objects
- ▶ Features modularize fragments from certain classes
- ▶ Fragment contains some methods/fields of a class belonging to one functionality
- ▶ Goals: code traceability, software customization

FOP Example

```
package util;
class Calc {
    void add() {
        e0 = e1 + e0;
        e1 = e2;
    }
}
```

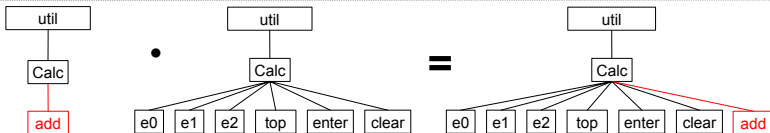
feature: Add

```
package util;
class Calc {
    int e0 = 0, e1 = 0,
        e2 = 0;
    void enter(int val) {
        e2 = e1; e1 = e0;
        e0 = val;
    }
    void clear() {
        e0 = e1 = e2 = 0;
    }
    String top() {
        return String.
            valueOf(e0);
    }
}
```

feature: CalcBase

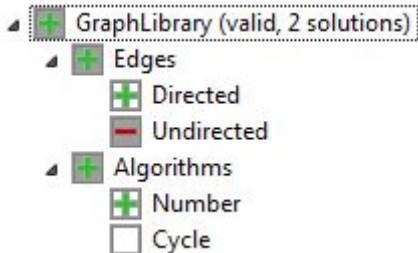
```
package util;
class Calc {
    int e0 = 0, e1 = 0,
        e2 = 0;
    void enter(int val) {
        e2 = e1; e1 = e0;
        e0 = val;
    }
    void clear() {
        e0 = e1 = e2 = 0;
    }
    String top() {
        //...
    }
    void add() {
        e0 = e1 + e0;
        e1 = e2;
    }
}
```

feature: CalcAdd



http://wwiti.cs.uni-magdeburg.de/iti_db/lehre/epmd/2009/slides/06_FOP.pdf

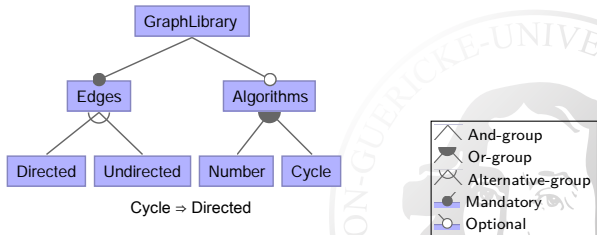
Configuration



- ▶ Selection of features
- ▶ Composition of features results in a program variant
- ▶ Not all combinations are useful

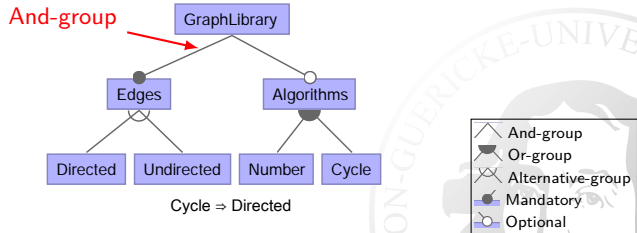
Feature Model

- ▶ Specifies valid combinations of features
- ▶ Graphically represented by a feature diagram
- ▶ Created for a particular domain
- ▶ Describes a software product line (SPL)



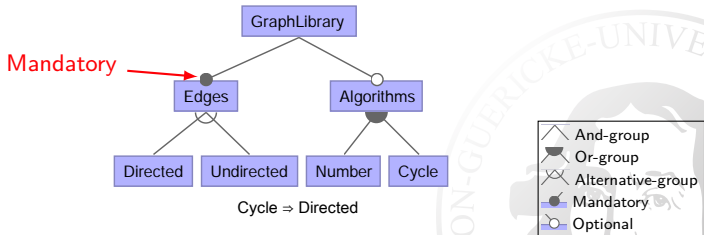
Feature Model

- Specifies valid combinations of features
- Graphically represented by a feature diagram
- Created for a particular domain
- Describes a software product line (SPL)



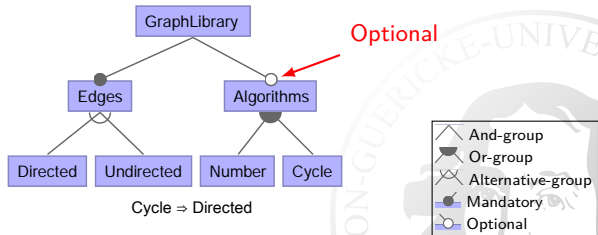
Feature Model

- ▶ Specifies valid combinations of features
- ▶ Graphically represented by a feature diagram
- ▶ Created for a particular domain
- ▶ Describes a software product line (SPL)



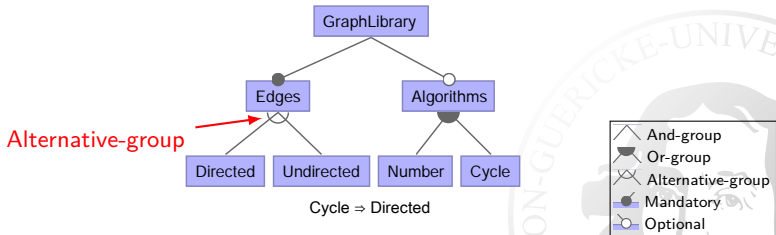
Feature Model

- ▶ Specifies valid combinations of features
- ▶ Graphically represented by a feature diagram
- ▶ Created for a particular domain
- ▶ Describes a software product line (SPL)



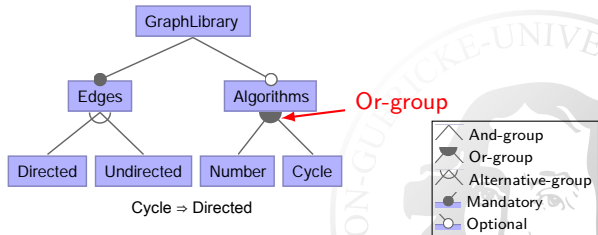
Feature Model

- ▶ Specifies valid combinations of features
- ▶ Graphically represented by a feature diagram
- ▶ Created for a particular domain
- ▶ Describes a software product line (SPL)



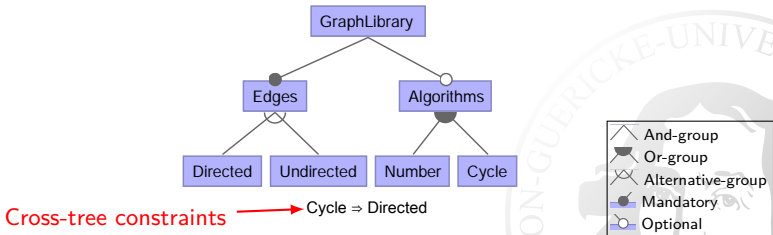
Feature Model

- Specifies valid combinations of features
- Graphically represented by a feature diagram
- Created for a particular domain
- Describes a software product line (SPL)



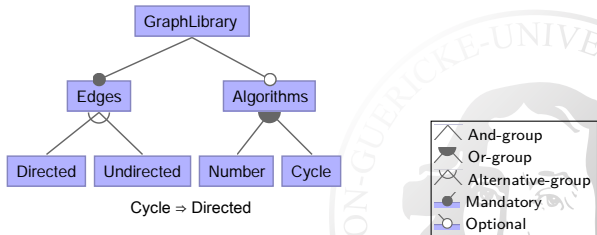
Feature Model

- Specifies valid combinations of features
- Graphically represented by a feature diagram
- Created for a particular domain
- Describes a software product line (SPL)



Feature Model

- ▶ Specifies valid combinations of features
- ▶ Graphically represented by a feature diagram
- ▶ Created for a particular domain
- ▶ Describes a software product line (SPL)

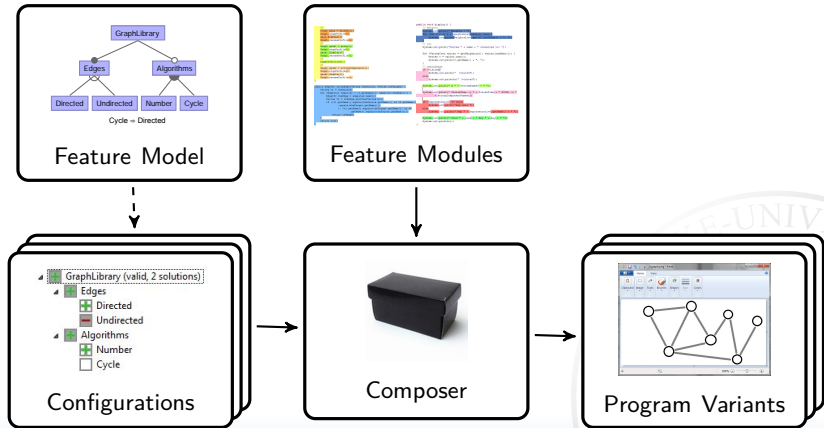


Composition Engines

Command-line tools used to compose files within FeatureIDE:

- ▶ AHEAD (jampack): .jak (Java 1.4)
<http://userweb.cs.utexas.edu/~schwartz/ATS.html>
- ▶ FeatureC++: .cpp (C++)
<http://www.fosd.de/fcpp>
- ▶ FeatureHouse: .java (Java 1.5), .cs (C#), .c/.h (C), .hs (Haskell), .jj (JavaCC), .als (Alloy), .xmi (UML)
<http://www.fosd.de/fh>

Feature-Oriented Software Development



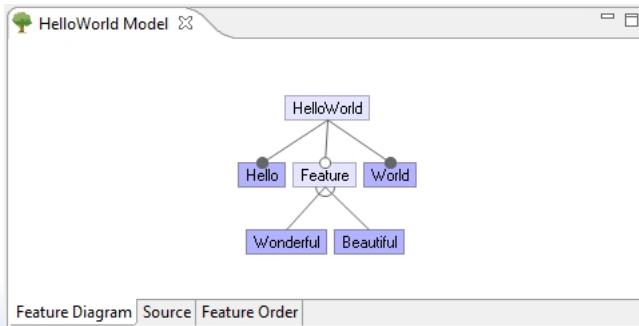
Content

- ▶ What is Feature-Oriented Software Development?
- ▶ What functionality does FeatureIDE provide?
 - ▶ Feature Model Editor + Edit View
 - ▶ Configuration Editor
 - ▶ Jak Editor
 - ▶ Collaboration Diagram
 - ▶ Feature Project Builder
 - ▶ Run Configurations
 - ▶ Creation Wizards
- ▶ How to start working with FeatureIDE?



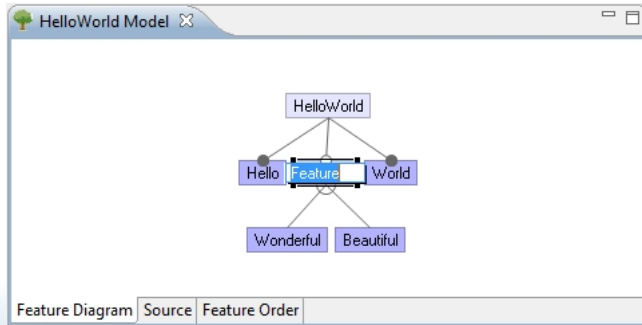
Feature Model Editor: Feature Diagram

- Double click to change connections and mandatory property



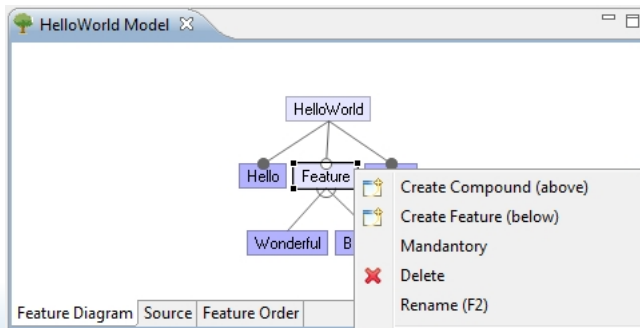
Feature Model Editor: Feature Diagram

- ▶ Double click to change connections and mandatory property
- ▶ Single click to rename features



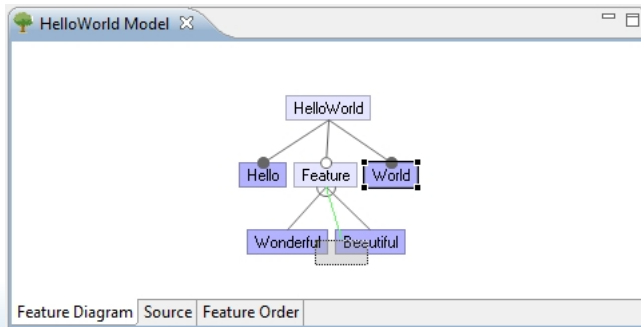
Feature Model Editor: Feature Diagram

- ▶ Double click to change connections and mandatory property
- ▶ Single click to rename features
- ▶ Right click to open context menu for features/connections



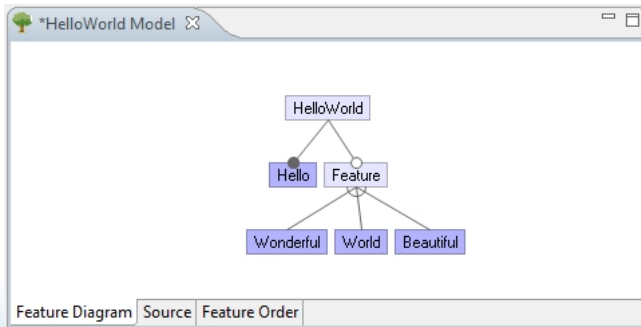
Feature Model Editor: Feature Diagram

- ▶ Double click to change connections and mandatory property
- ▶ Single click to rename features
- ▶ Right click to open context menu for features/connections
- ▶ Drag



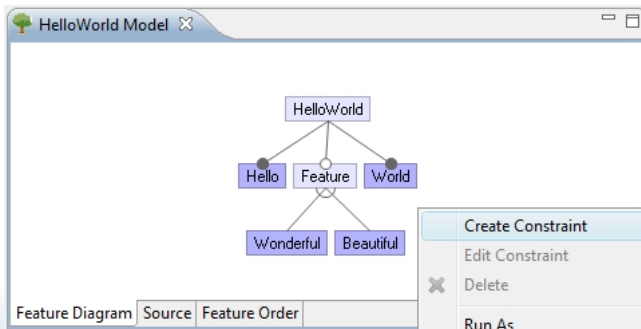
Feature Model Editor: Feature Diagram

- ▶ Double click to change connections and mandatory property
- ▶ Single click to rename features
- ▶ Right click to open context menu for features/connections
- ▶ Drag and drop features



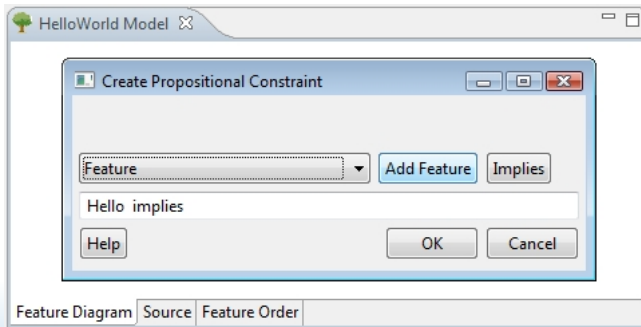
Feature Model Editor: Feature Diagram

- ▶ Double click to change connections and mandatory property
- ▶ Single click to rename features
- ▶ Right click to open context menu for features/connections
- ▶ Drag and drop features
- ▶ Context menu



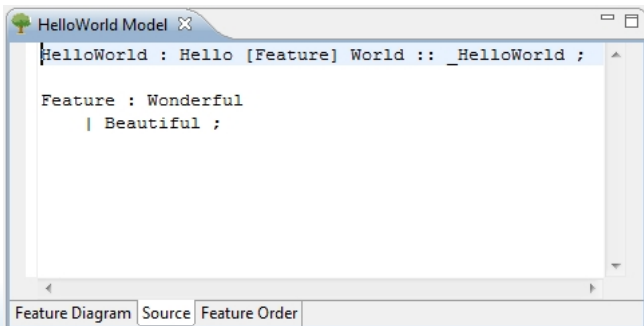
Feature Model Editor: Feature Diagram

- ▶ Double click to change connections and mandatory property
- ▶ Single click to rename features
- ▶ Right click to open context menu for features/connections
- ▶ Drag and drop features
- ▶ Context menu to open Constraint Editor



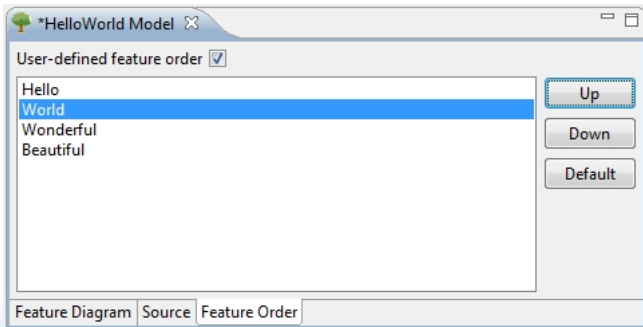
Feature Model Editor - Grammar

- ▶ Source tab contains the GUIDSL grammar representation
- ▶ [] - optional feature
- ▶ | - Or-group -or- Alternative-group depending on parent
- ▶ + - mandatory feature and Or-group below
- ▶ * - optional feature and Or-group below



Feature Model Editor - Feature Order

- ▶ Order of features matters: can influence program behavior
- ▶ Default order: pre-order traversal of the feature diagram
- ▶ User-defined order possible
- ▶ Applies to all configurations



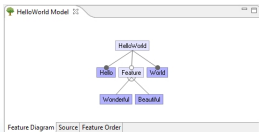
Feature Model Editor - Synchronization

Before saving:

- ▶ When switching tab, changes are propagated

When saving:

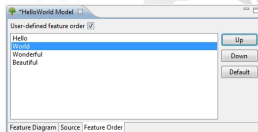
- ▶ Feature folders are created, removed, and renamed
- ▶ Updating order of features in configurations
- ▶ Checking which configurations are valid/invalid
- ▶ Current content of Configuration Editor updated



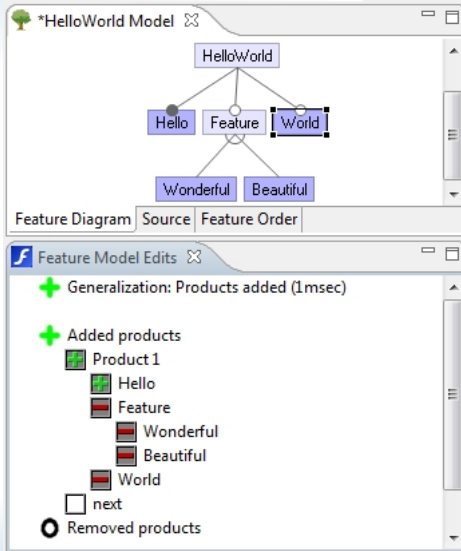
The screenshot shows a window titled 'HelloWorld Model'. It displays the source code of the model in the 'Source' tab. The code is as follows:

```
HelloWorld : Hello [Feature] World :: _HelloWorld ;  
Feature : Wonderful  
| Beautiful ;
```

The bottom of the window has a tab bar with 'Feature Diagram', 'Source', and 'Feature Order'.

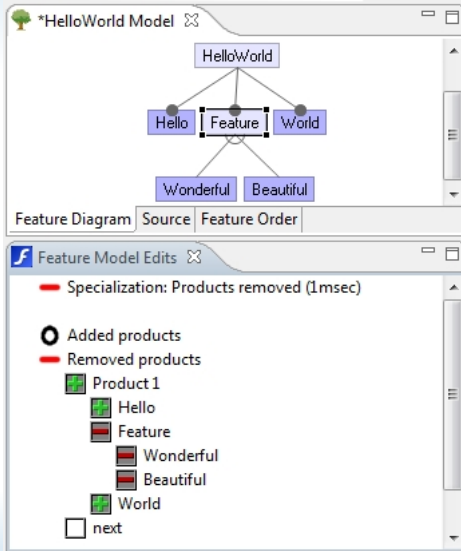


Feature Model Edit View



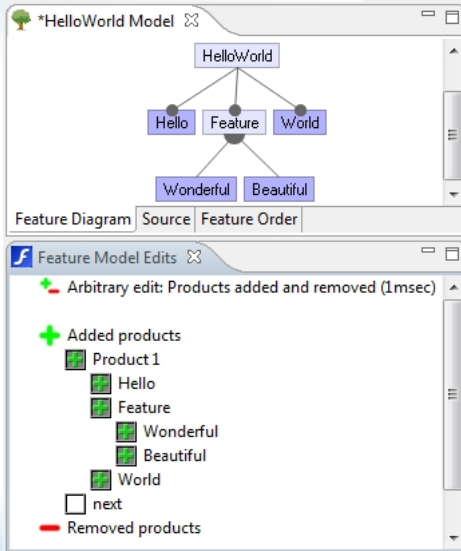
- ▶ Compares current edited feature model with last saved version
- ▶ Result: variants were **added**, removed, both, or none
- ▶ Provides examples for **added**/removed variants
- ▶ Statistical data for both versions of the feature model

Feature Model Edit View



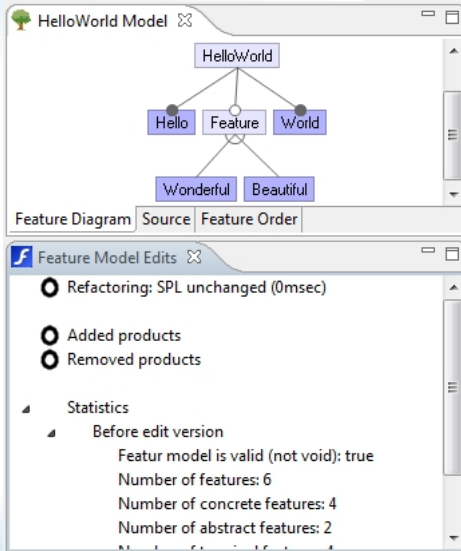
- ▶ Compares current edited feature model with last saved version
- ▶ Result: variants were added, **removed**, both, or none
- ▶ Provides examples for added/**removed** variants
- ▶ Statistical data for both versions of the feature model

Feature Model Edit View



- ▶ Compares current edited feature model with last saved version
- ▶ Result: variants were added, removed, **both**, or none
- ▶ Provides examples for **added/removed** variants
- ▶ Statistical data for both versions of the feature model

Feature Model Edit View



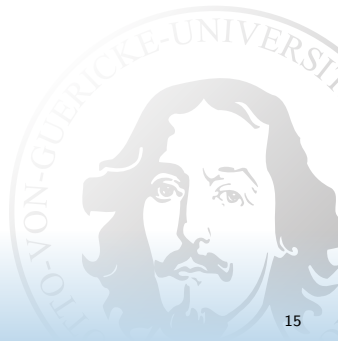
- ▶ Compares current edited feature model with last saved version
- ▶ Result: variants were added, removed, both, or **none**
- ▶ Provides examples for added/removed variants
- ▶ Statistical data for both versions of the feature model

Configuration Editor

► ...

► ...

► ...

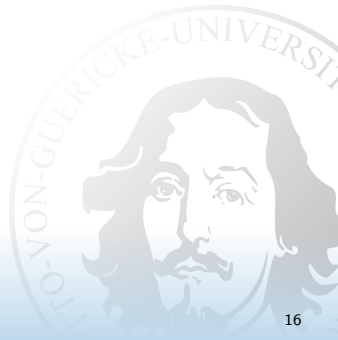


Jak Editor

▶ ...

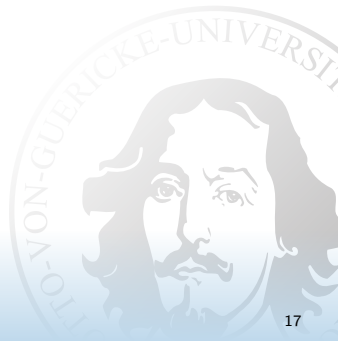
▶ ...

▶ ...



Collaboration Diagram

- ▶ ...
- ▶ ...
- ▶ ...



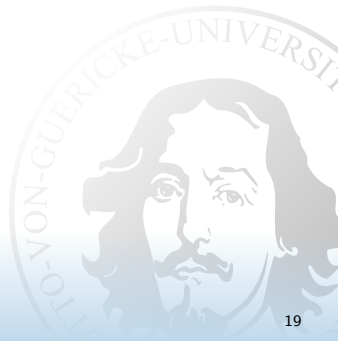
Feature Project Builder

- ▶ ...
- ▶ ...
- ▶ ...



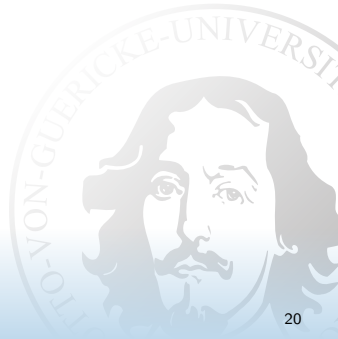
Run Configurations

- ▶ ...
- ▶ ...
- ▶ ...



Creation Wizards

- ▶ ...
- ▶ ...
- ▶ ...



Content

- ▶ What is Feature-Oriented Software Development?
- ▶ What functionality does FeatureIDE provide?
- ▶ How to start working with FeatureIDE?
 - ▶ FeatureIDE installation
 - ▶ FeatureIDE project structure
 - ▶ Cheat sheet

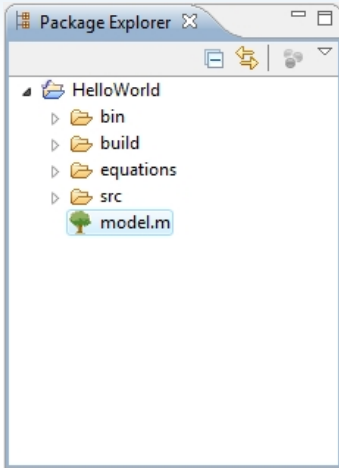


Installation of FeatureIDE

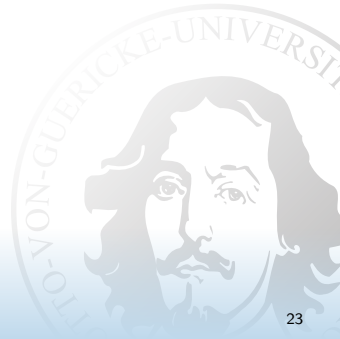
- ▶ ...
- ▶ ...
- ▶ ...



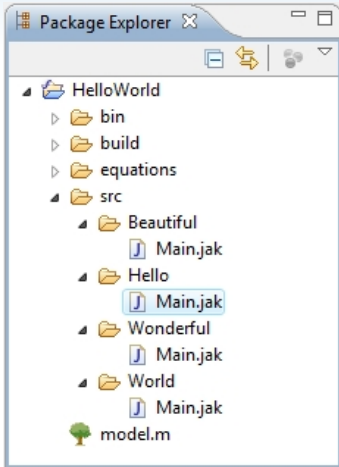
FeatureIDE Project Structure



- Feature model file in the GUIDSL-format



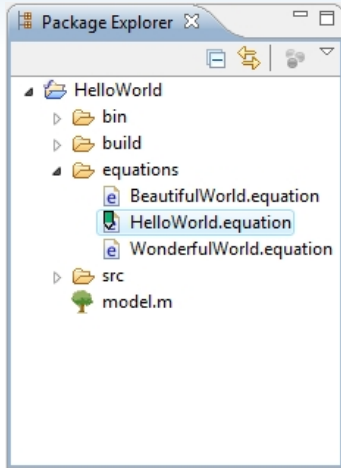
FeatureIDE Project Structure



- ▶ Feature model file in the GUIDSL-format
- ▶ Source folder containing a folder for every feature including files to compose

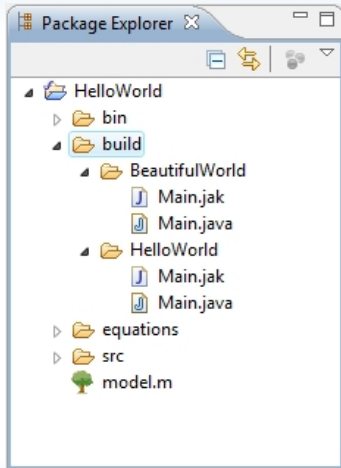


FeatureIDE Project Structure



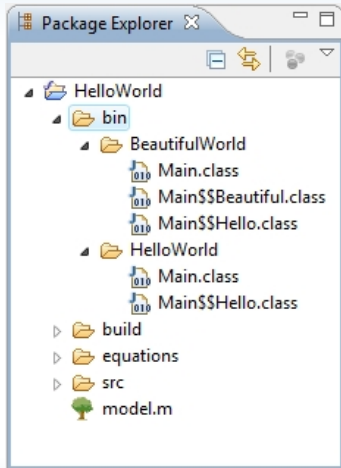
- ▶ Feature model file in the GUIDSL-format
- ▶ Source folder containing a folder for every feature including files to compose
- ▶ Configurations containing selected features from the feature model

FeatureIDE Project Structure



- ▶ Feature model file in the GUIDSL-format
- ▶ Source folder containing a folder for every feature including files to compose
- ▶ Configurations containing selected features from the feature model
- ▶ Composed source files for several configurations (might be helpful when debugging)

FeatureIDE Project Structure



- ▶ Feature model file in the GUIDSL-format
- ▶ Source folder containing a folder for every feature including files to compose
- ▶ Configurations containing selected features from the feature model
- ▶ Composed source files for several configurations (might be helpful when debugging)
- ▶ Binary files for several configurations

Cheat Sheet

▶ ...

▶ ...

▶ ...

