OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

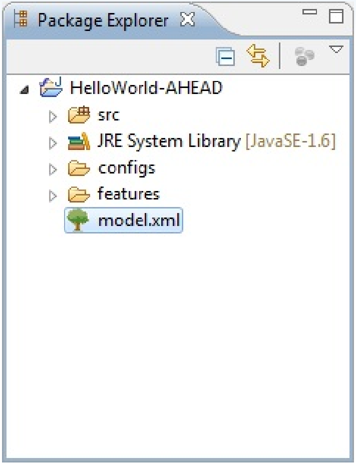
INF FAKULTÄT FÜR INFORMATIK

FeatureIDE: Overview

Thomas Thüm, Jens Meinicke

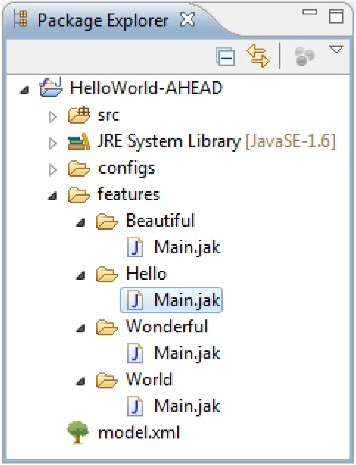
October 23, 2015

FeatureIDE Project Structure



- Feature model file in a supported format (default: xml)

FeatureIDE Project Structure

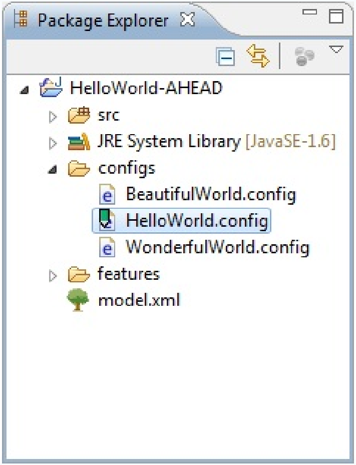


Package Explorer

- HelloWorld-AHEAD
  - src
  - JRE System Library [JavaSE-1.6]
  - configs
  - features
    - Beautiful
      - Main.jak
    - Hello
      - Main.jak
    - Wonderful
      - Main.jak
    - World
      - Main.jak
  - model.xml

- ▶ Feature model file in a supported format (default: xml)
- ▶ Source folder containing a folder for every feature including files to compose

FeatureIDE Project Structure

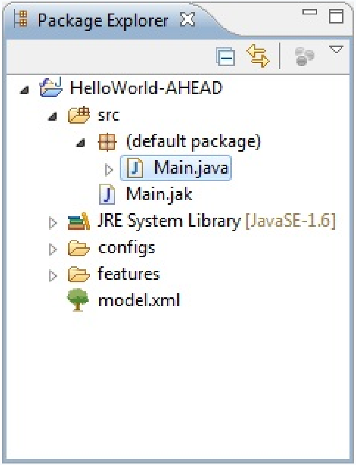


Package Explorer

- HelloWorld-AHEAD
  - src
  - JRE System Library [JavaSE-1.6]
  - configs
    - BeautifulWorld.config
    - HelloWorld.config
    - WonderfulWorld.config
  - features
  - model.xml

- Feature model file in a supported format (default: xml)
- Source folder containing a folder for every feature including files to compose
- Configurations containing selected features from the feature model

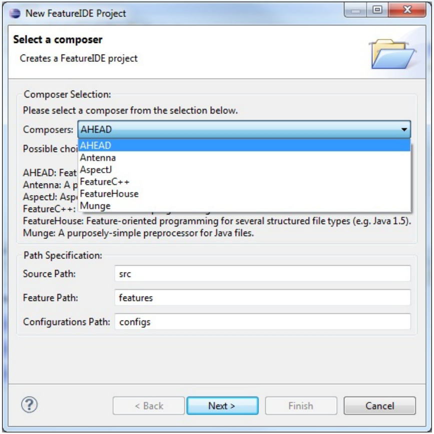
FeatureIDE Project Structure


Package Explorer

- HelloWorld-AHEAD
  - src
    - (default package)
      - Main.java
    - Main.jak
  - JRE System Library [JavaSE-1.6]
  - configs
  - features
  - model.xml

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

Creation Wizards



Select a composer

Creates a FeatureIDE project

Composer Selection:

Please select a composer from the selection below.

Composers: AHEAD

Possible choi

AHEAD: Feat

Antenna: A p

AspectJ: Asp

FeatureC++:

FeatureHouse: Feature-oriented programming for several structured file types (e.g. Java 1.5).

Munge: A purposely-simple preprocessor for Java files.

Path Specification:

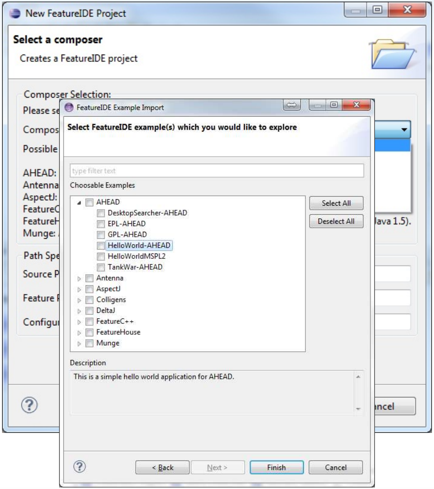
Source Path: src

Feature Path: features

Configurations Path: configs

- FeatureIDE project wizard: creates basic feature model and project structure, also can add the FeatureIDE Nature to a project at your workspace

Creation Wizards


New FeatureIDE Project

Select a composer

Creates a FeatureIDE project

FeatureIDE Example Import

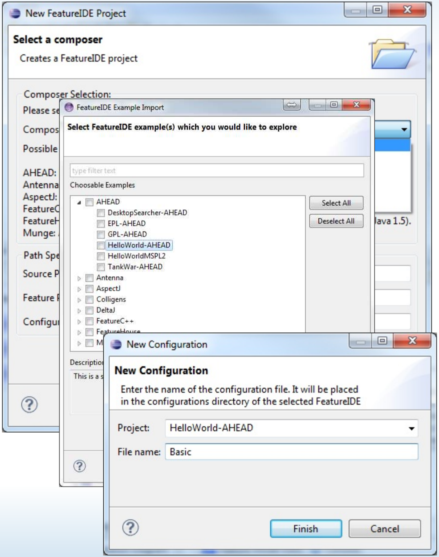
Select FeatureIDE example(s) which you would like to explore

Description

This is a simple hello world application for AHEAD.

- FeatureIDE project wizard: creates basic feature model and project structure, also can add the FeatureIDE Nature to a project at your workspace
- Example project wizard

Creation Wizards



New FeatureIDE Project

Select a composer

Creates a FeatureIDE project

Composer Selection:

Please se

Compos

Possible

AHEAD:
Antenna
AspectJ:
FeatureC
FeatureH
Munge:

Java 1.5).

Path Spe

Source P

Feature P

Configu

FeatureIDE Example Import

Select FeatureIDE example(s) which you would like to explore

type filter text

Choosable Examples

- □ AHEAD
  - □ DesktopSearcher-AHEAD
  - □ EPL-AHEAD
  - □ GPL-AHEAD
  - □ HelloWorld-AHEAD
  - □ HelloWorldMSPL2
  - □ TankWar-AHEAD
- □ Antenna
- □ AspectJ
- □ Colligens
- □ DeltaJ
- □ FeatureC++
- □ FeatureHouse
- □ M

Description

This is a s

New Configuration

New Configuration

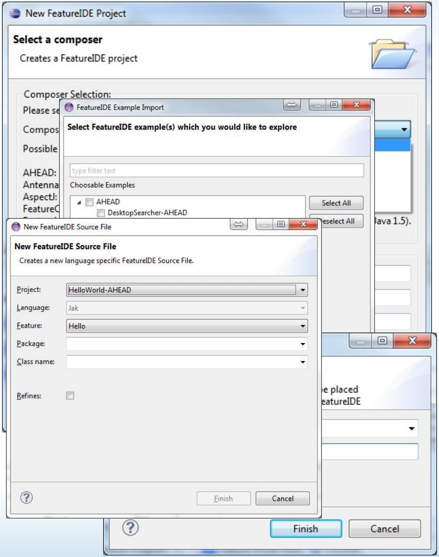
Enter the name of the configuration file. It will be placed in the configurations directory of the selected FeatureIDE

Project: HelloWorld-AHEAD

File name: Basic

- FeatureIDE project wizard: creates basic feature model and project structure, also can add the FeatureIDE Nature to a project at your workspace
- Example project wizard
- Configuration file wizard

Creation Wizards



New FeatureIDE Project

Select a composer

Creates a FeatureIDE project

Composer Selection:

Please se

Compos

Possible

AHEAD:
Antenna
AspectJ:
FeatureC

Java 1.5).

FeatureIDE Example Import

Select FeatureIDE example(s) which you would like to explore

type filter text

Choosable Examples

- □ AHEAD
  - □ DesktopSearcher-AHEAD

Select All

Deselect All

New FeatureIDE Source File

New FeatureIDE Source File

Creates a new language specific FeatureIDE Source File.

Project: HelloWorld-AHEAD

Language: Jak

Feature: Hello

Package:

Class name:

Refines: □

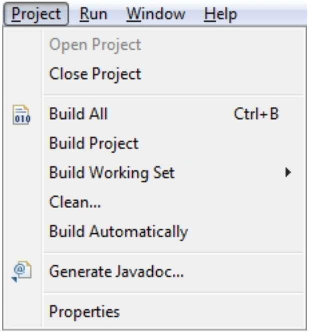
Finish Cancel

e placed
eatureIDE

Finish Cancel

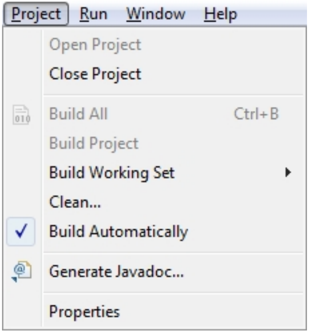
- FeatureIDE project wizard: creates basic feature model and project structure, also can add the FeatureIDE Nature to a project at your workspace
- Example project wizard
- Configuration file wizard
- FeatureIDE source file wizard

FeatureIDE Project Builder



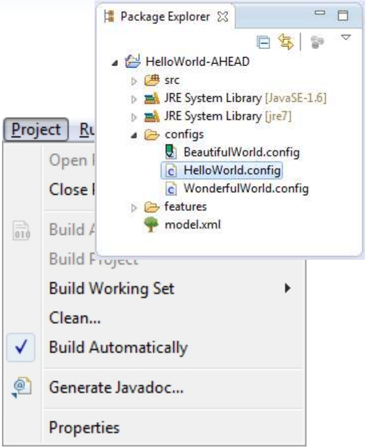
- Specific to composition engine
- Build: compose features contained in current configuration
- Clean: removing all resources in bin/build folder
- Build automatically: based on changes in source folder or current configuration

FeatureIDE Project Builder



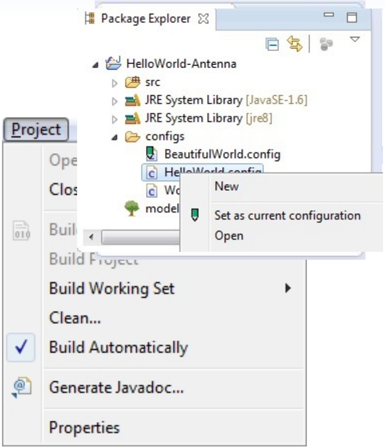
- Specific to composition engine
- Build: compose features contained in current configuration
- Clean: removing all resources in bin/build folder
- Build automatically: based on changes in source folder or current configuration

FeatureIDE Project Builder



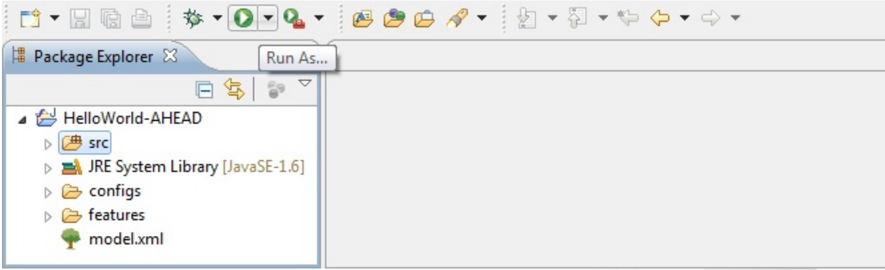
- Specific to composition engine
- Build: compose features contained in current configuration
- Clean: removing all resources in bin/build folder
- Build automatically: based on changes in source folder or current configuration

FeatureIDE Project Builder



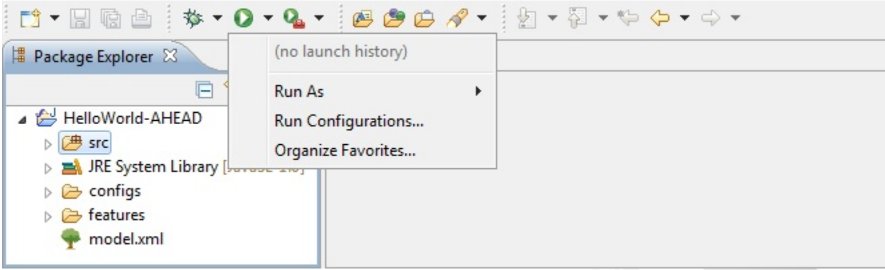
- Specific to composition engine
- Build: compose features contained in current configuration
- Clean: removing all resources in bin/build folder
- Build automatically: based on changes in source folder or current configuration

Run Configurations



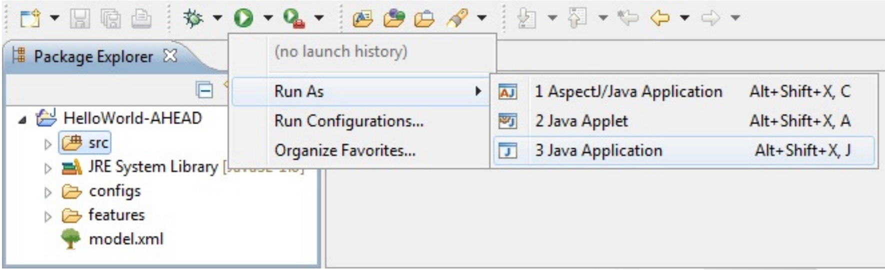
- The project has other natures (e.g. Java nature) next to the FeatureIDE nature, so you can run your application in the common way

Run Configurations



- The project has other natures (e.g. Java nature) next to the FeatureIDE nature, so you can run your application in the common way

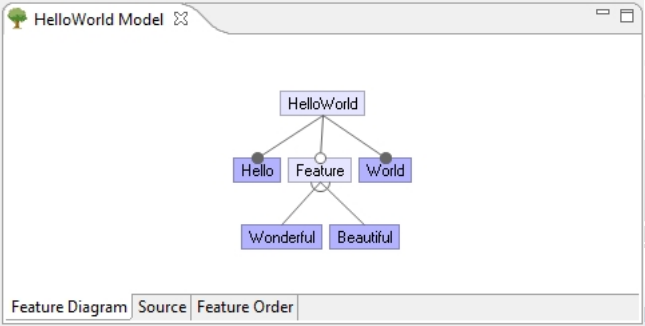
Run Configurations



- The project has other natures (e.g. Java nature) next to the FeatureIDE nature, so you can run your application in the common way

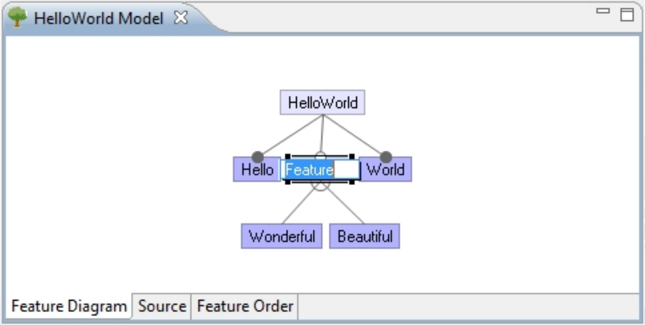
Feature Model Editor - Feature Diagram

- Double click to change connections and mandatory property



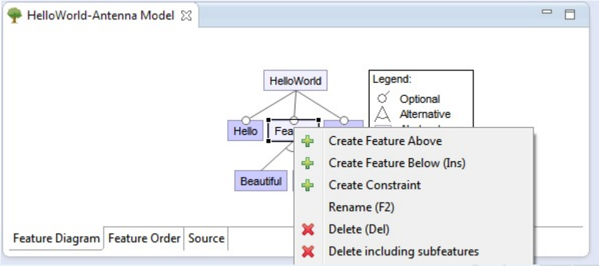
Feature Model Editor - Feature Diagram

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



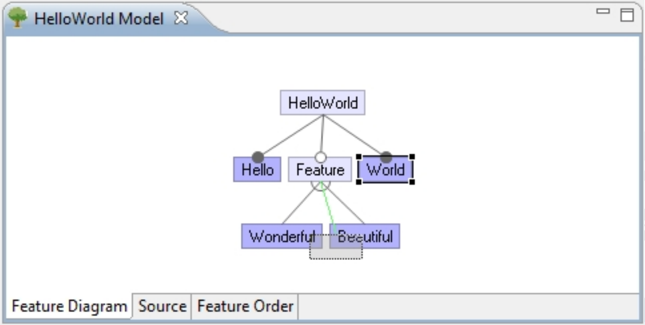
Feature Model Editor - Feature Diagram

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



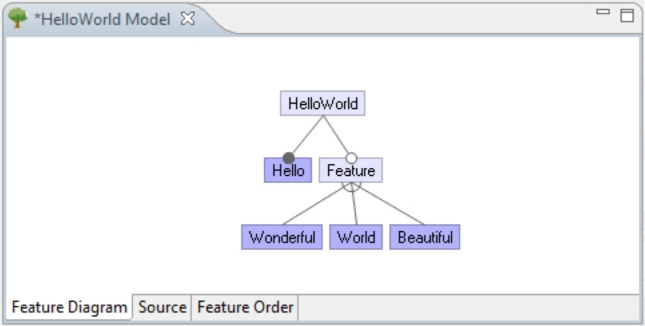
Feature Model Editor - Feature Diagram

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



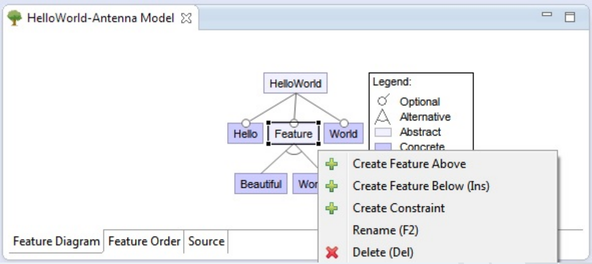
Feature Model Editor - Feature Diagram

- 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



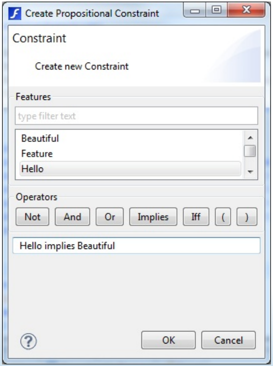
Feature Model Editor - Feature Diagram

- 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



Create Propositional Constraint

Constraint

Create new Constraint

Features

type filter text

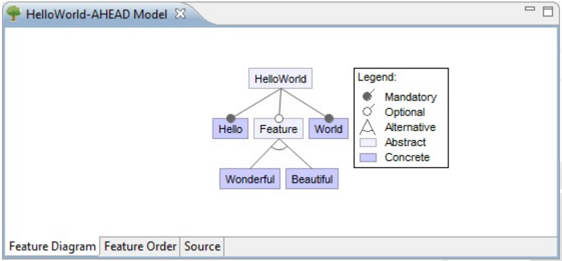
Beautiful
Feature
Hello

Operators

Hello implies Beautiful

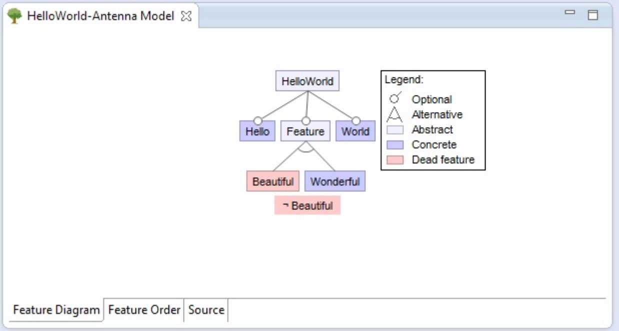
Feature Model Editor - Feature Diagram

- Legend for the feature model elements and colors



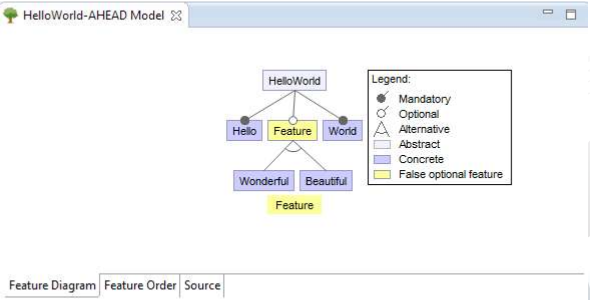
Feature Model Editor - Feature Diagram

- Legend for the feature model elements and colors
- Highlighting dead features



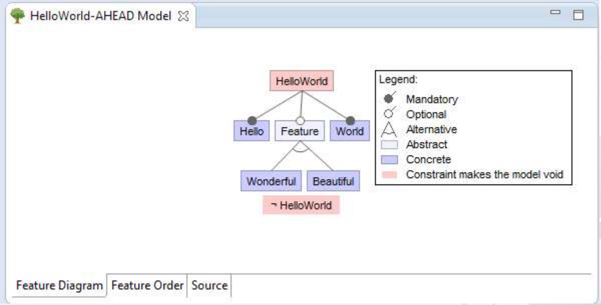
Feature Model Editor - Feature Diagram

- Legend for the feature model elements and colors
- Highlighting dead features
- Highlighting false optional features



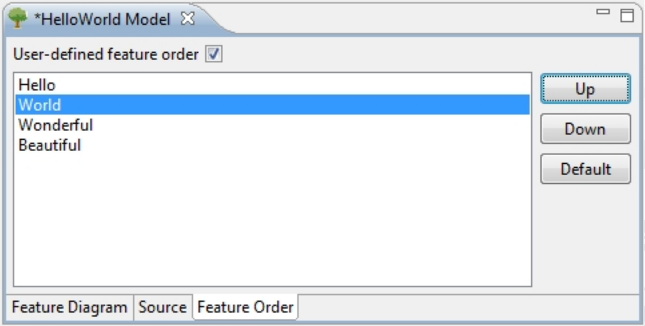
Feature Model Editor - Feature Diagram

- Legend for the feature model elements and colors
- Highlighting dead features
- Highlighting false optional features
- Highlight if the model is void



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



*HelloWorld Model

User-defined feature order ☑

Hello
World
Wonderful
Beautiful

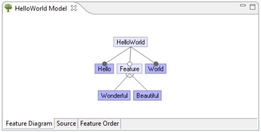
Feature Model Editor - Synchronization

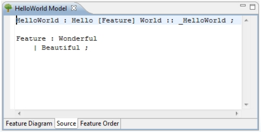
Before saving:

- When switching tab, changes are propagated

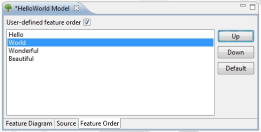
When saving:

- 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





```
HelloWorld : Hello [Feature] World :: _HelloWorld ;

Feature : Wonderful
    | Beautiful ;
```

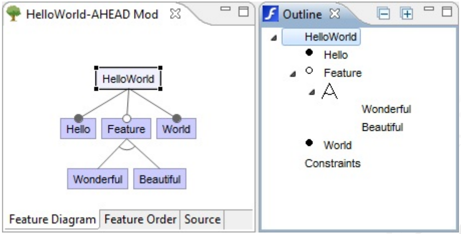


User-defined feature order ☑

Hello
World
Wonderful
Beautiful

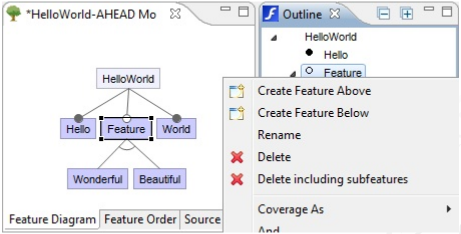
FeatureIDE Outline View: Feature Model

- Tree representation of the Feature Model



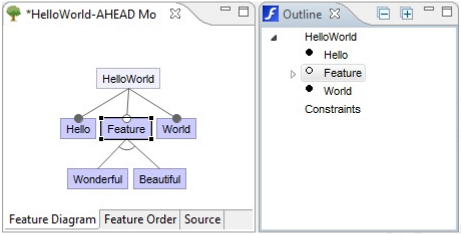
FeatureIDE Outline View: Feature Model

- Tree representation of the Feature Model
- Same context menu like at the editor



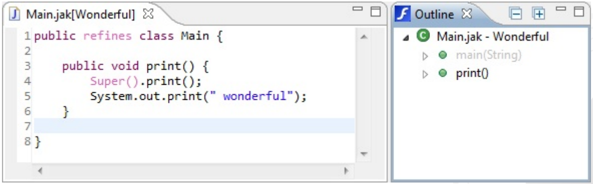
FeatureIDE Outline View: Feature Model

- Tree representation of the Feature Model
- Same context menu like at the editor
- Minimize subtrees for a better overview



FeatureIDE Collaboration Outline

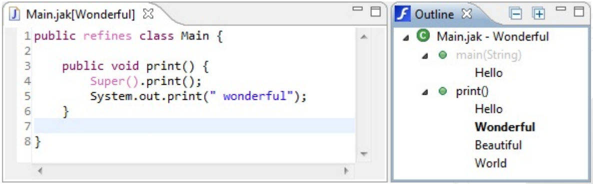
- Additional to the normal outline, it contains all methods of this class from other features



```
public refines class Main {

    public void print() {
        Super().print();
        System.out.print(" wonderful");
    }

}
```

FeatureIDE Collaboration Outline

- Additional to the normal outline, it contains all methods of this class from other features
- The subtree of a method contains all features implementing it

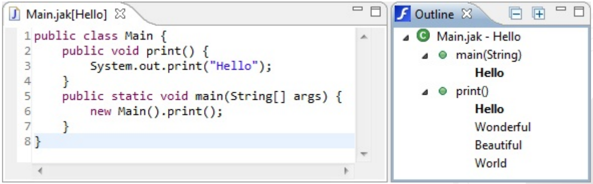


```
public refines class Main {

    public void print() {
        Super().print();
        System.out.print(" wonderful");
    }

}
```

- Main.jak - Wonderful
  - main(String)
    - Hello
  - print()
    - Hello
    - **Wonderful**
    - Beautiful
    - World

FeatureIDE Collaboration Outline

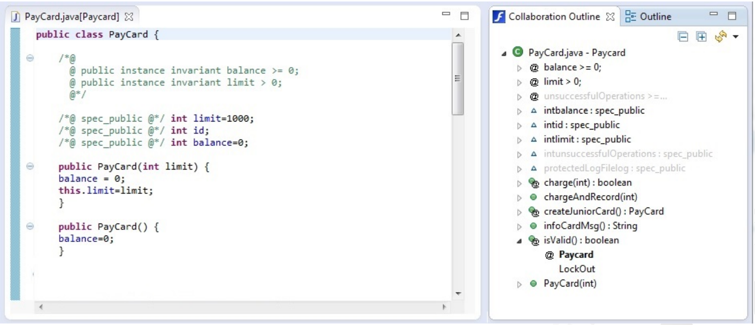
- Additional to the normal outline, it contains all methods of this class from other features
- The subtree of a method contains all features implementing it
- Clicking the feature will open the corresponding file



```
public class Main {
    public void print() {
        System.out.print("Hello");
    }
    public static void main(String[] args) {
        new Main().print();
    }
}
```

- Main.jak - Hello
  - main(String)
    - **Hello**
  - print()
    - **Hello**
    - Wonderful
    - Beautiful
    - World

FeatureIDE Collaboration Outline: Contracts

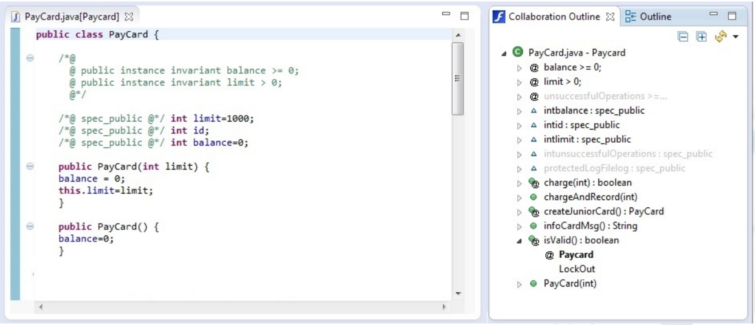


```
public class PayCard {

    /*@
      @ public instance invariant balance >= 0;
      @ public instance invariant limit > 0;
      @*/

    /*@ spec_public @*/ int limit=1000;
    /*@ spec_public @*/ int id;
    /*@ spec_public @*/ int balance=0;

    public PayCard(int limit) {
    balance = 0;
    this.limit=limit;
    }

    public PayCard() {
    balance=0;
    }
```

```
PayCard.java - Paycard
    balance >= 0;
    limit > 0;
    unsuccessfulOperations >=...
    intbalance : spec_public
    intid : spec_public
    intlimit : spec_public
    intunsuccessfulOperations : spec_public
    protectedLogFilelog : spec_public
    charge(int) : boolean
    chargeAndRecord(int)
    createJuniorCard() : PayCard
    infoCardMsg() : String
    isValid() : boolean
        Paycard
        LockOut
    PayCard(int)
```

- Class invariants are shown in Outline above class fields

FeatureIDE Collaboration Outline: Contracts

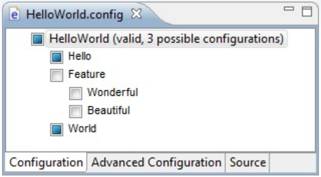


```
public class PayCard {

    /*@
      @ public instance invariant balance >= 0;
      @ public instance invariant limit > 0;
      @*/

    /*@ spec_public @*/ int limit=1000;
    /*@ spec_public @*/ int id;
    /*@ spec_public @*/ int balance=0;

    public PayCard(int limit) {
    balance = 0;
    this.limit=limit;
    }

    public PayCard() {
    balance=0;
    }
```

```
PayCard.java - Paycard
    @ balance >= 0;
    @ limit > 0;
    @ unsuccessfulOperations >=...
    intbalance : spec_public
    intid : spec_public
    intlimit : spec_public
    intunsuccessfulOperations : spec_public
    protectedLogFilelog : spec_public
    charge(int) : boolean
    chargeAndRecord(int)
    createJuniorCard() : PayCard
    infoCardMsg() : String
    isValid() : boolean
        @ Paycard
          LockOut
    PayCard(int)
```

- Class invariants are shown in Outline above class fields
- Contracts of methods are displayed by decorating @-icon
- Features in subtrees of a method with additional @-icon if contract exists

Configuration Editor: Default Page

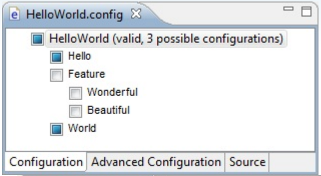


HelloWorld.config

- ■ HelloWorld (valid, 3 possible configurations)
  - ■ Hello
  - □ Feature
    - □ Wonderful
    - □ Beautiful
  - ■ World

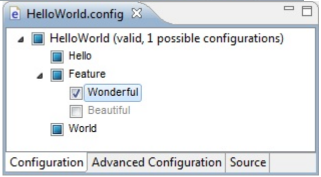
- Manual select or deselect the features
- Automatic decisions based on current manual decisions
- Is current configuration valid?
- Counting possible configurations
- Highlightings to find a valid configuration (blue: deselect, green: select the feature)

Configuration Editor: Default Page



HelloWorld.config

- ■ HelloWorld (valid, 3 possible configurations)
  - ■ Hello
  - □ Feature
    - □ Wonderful
    - □ Beautiful
  - ■ World

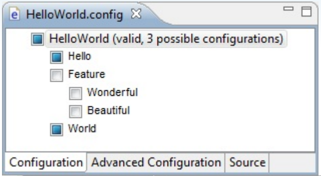


HelloWorld.config

- ■ HelloWorld (valid, 1 possible configurations)
  - ■ Hello
  - ■ Feature
    - ☑ Wonderful
    - □ Beautiful
  - ■ World

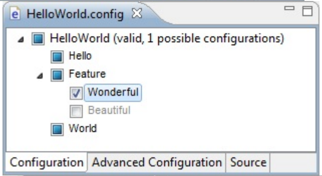
- Manual select or deselect the features
- Automatic decisions based on current manual decisions
- Is current configuration valid?
- Counting possible configurations
- Highlightings to find a valid configuration (blue: deselect, green: select the feature)

Configuration Editor: Default Page



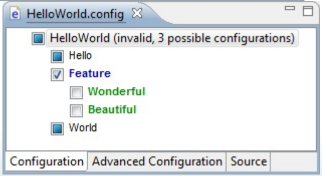
HelloWorld.config

- ■ HelloWorld (valid, 3 possible configurations)
  - ■ Hello
  - □ Feature
    - □ Wonderful
    - □ Beautiful
  - ■ World



HelloWorld.config

- ■ HelloWorld (valid, 1 possible configurations)
  - ■ Hello
  - ■ Feature
    - ☑ Wonderful
    - □ Beautiful
  - ■ World

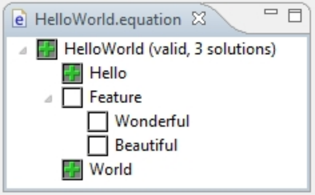


HelloWorld.config

- ■ HelloWorld (invalid, 3 possible configurations)
  - ■ Hello
  - ☑ Feature
    - □ Wonderful
    - □ Beautiful
  - ■ World

- Manual select or deselect the features
- Automatic decisions based on current manual decisions
- Is current configuration valid?
- Counting possible configurations
- Highlightings to find a valid configuration (blue: deselect, green: select the feature)

Configuration Editor: Advanced Page



HelloWorld.equation

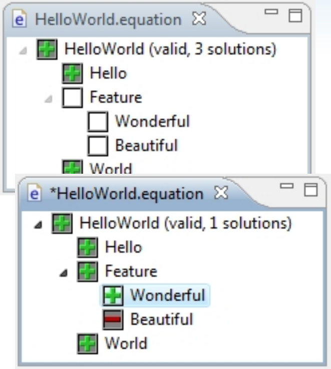
- ⊞ HelloWorld (valid, 3 solutions)
  - ⊞ Hello
  - □ Feature
    - □ Wonderful
    - □ Beautiful
  - ⊞ World

- Additional to the default page you can eliminate features

On save:

- Error marker if invalid
- Build process started if current configuration

Configuration Editor: Advanced Page



HelloWorld.equation

- ☑ HelloWorld (valid, 3 solutions)
  - ☑ Hello
  - □ Feature
    - □ Wonderful
    - □ Beautiful
  - ☑ World

*HelloWorld.equation

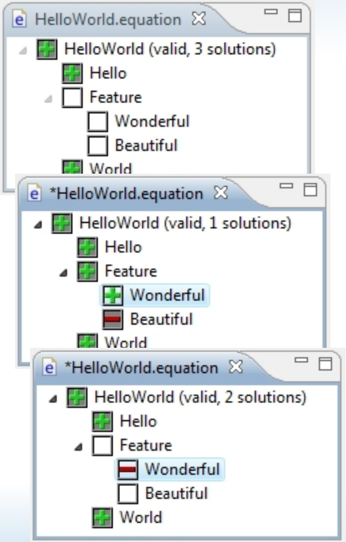
- ☑ HelloWorld (valid, 1 solutions)
  - ☑ Hello
  - ☑ Feature
    - ☑ Wonderful
    - ☒ Beautiful
  - ☑ World

- Additional to the default page you can eliminate features

On save:

- Error marker if invalid
- Build process started if current configuration

Configuration Editor: Advanced Page



HelloWorld.equation

- ☑ HelloWorld (valid, 3 solutions)
  - ☑ Hello
  - □ Feature
    - □ Wonderful
    - □ Beautiful
  - ☑ World

*HelloWorld.equation

- ☑ HelloWorld (valid, 1 solutions)
  - ☑ Hello
  - ☑ Feature
    - ☑ Wonderful
    - ☒ Beautiful
  - ☑ World

*HelloWorld.equation

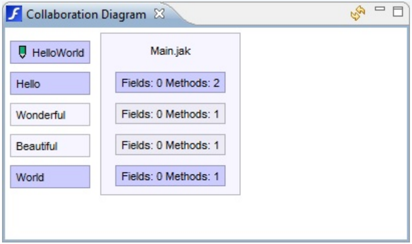
- ☑ HelloWorld (valid, 2 solutions)
  - ☑ Hello
  - □ Feature
    - ☒ Wonderful
    - □ Beautiful
  - ☑ World

- Additional to the default page you can eliminate features

On save:

- Error marker if invalid
- Build process started if current configuration

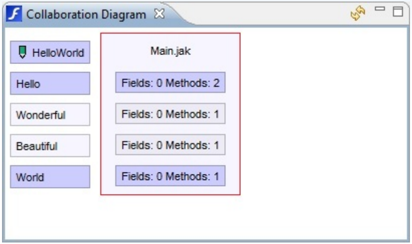
Collaboration Diagram: Introduction



HelloWorld	Main.jak
Hello	Fields: 0 Methods: 2
Wonderful	Fields: 0 Methods: 1
Beautiful	Fields: 0 Methods: 1
World	Fields: 0 Methods: 1

- Visualization of a software product line implementation

Collaboration Diagram: Introduction

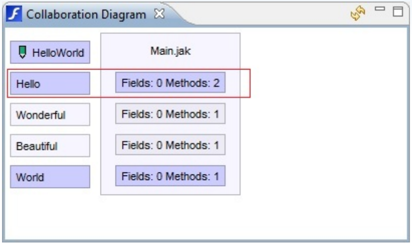


Collaboration Diagram

	Main.jak
HelloWorld	
Hello	Fields: 0 Methods: 2
Wonderful	Fields: 0 Methods: 1
Beautiful	Fields: 0 Methods: 1
World	Fields: 0 Methods: 1

- Visualization of a software product line implementation
- Class - a composed class

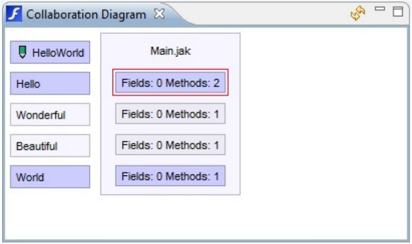
Collaboration Diagram: Introduction



HelloWorld	Main.jak
Hello	Fields: 0 Methods: 2
Wonderful	Fields: 0 Methods: 1
Beautiful	Fields: 0 Methods: 1
World	Fields: 0 Methods: 1

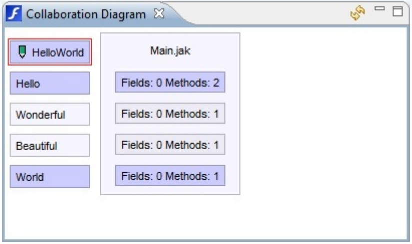
- Visualization of a software product line implementation
- Class - a composed class
- Collaboration - set of classes to implement a feature

Collaboration Diagram: Introduction



- Visualization of a software product line implementation
- Class - a composed class
- Collaboration - set of classes to implement a feature
- Role - a class fragment to implement a feature of a class
- A lower role will refine the functionality of an upper one

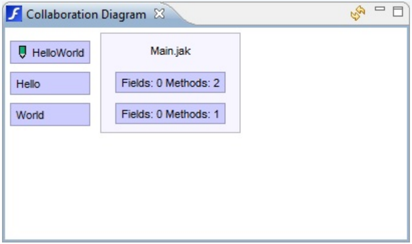
Collaboration Diagram: Introduction


Collaboration Diagram

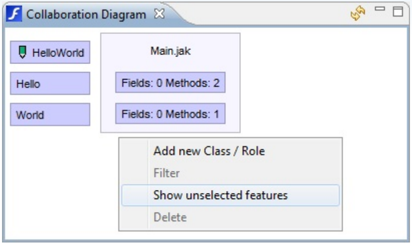
HelloWorld	Main.jak
Hello	Fields: 0 Methods: 2
Wonderful	Fields: 0 Methods: 1
Beautiful	Fields: 0 Methods: 1
World	Fields: 0 Methods: 1

- Visualization of a software product line implementation
- Class - a composed class
- Collaboration - set of classes to implement a feature
- Role - a class fragment to implement a feature of a class
- A lower role will refine the functionality of an upper one
- Configuration - the corresponding configuration

Collaboration Diagram: Functionality

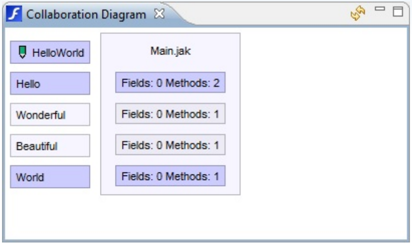


Collaboration Diagram: Functionality



- Also showing unselected features

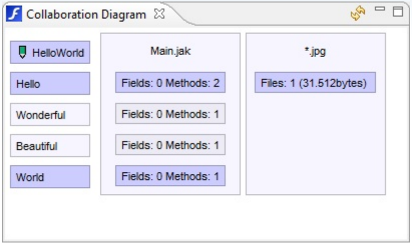
Collaboration Diagram: Functionality



HelloWorld	Main.jak
Hello	Fields: 0 Methods: 2
Wonderful	Fields: 0 Methods: 1
Beautiful	Fields: 0 Methods: 1
World	Fields: 0 Methods: 1

- Also showing unselected features

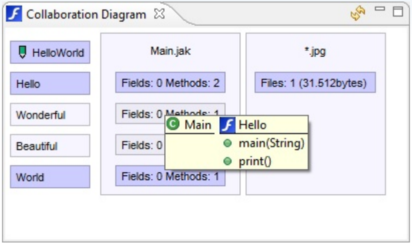
Collaboration Diagram: Functionality



	Main.jak	*.jpg
HelloWorld		
Hello	Fields: 0 Methods: 2	Files: 1 (31.512bytes)
Wonderful	Fields: 0 Methods: 1	
Beautiful	Fields: 0 Methods: 1	
World	Fields: 0 Methods: 1	

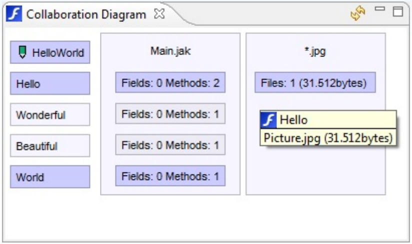
- Also showing unselected features
- Occurence of arbitrary files

Collaboration Diagram: Functionality



- Also showing unselected features
- Occurence of arbitrary files
- Tooltip containing all methods and fiels of the role

Collaboration Diagram: Functionality



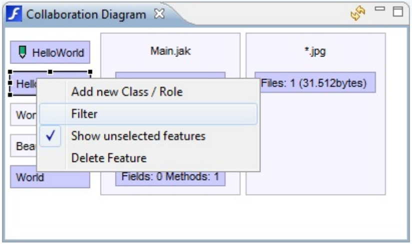
Collaboration Diagram

HelloWorld	Main.jak	*.jpg
Hello	Fields: 0 Methods: 2	Files: 1 (31.512bytes)
Wonderful	Fields: 0 Methods: 1	
Beautiful	Fields: 0 Methods: 1	
World	Fields: 0 Methods: 1	

Hello
Picture.jpg (31.512bytes)

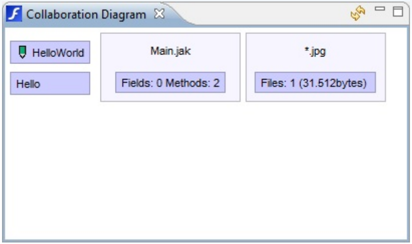
- Also showing unselected features
- Occurence of arbitrary files
- Tooltip containing all methods and fiels of the role
- Tooltip containing all arbitrary files of the same type

Collaboration Diagram: Functionality



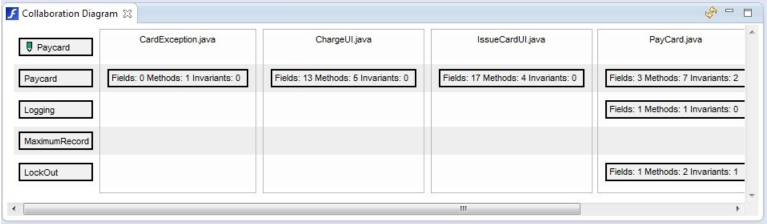
- Also showing unselected features
- Occurence of arbitrary files
- Tooltip containing all methods and fiels of the role
- Tooltip containing all arbitrary files of the same type
- Filter the diagram by a feature or class

Collaboration Diagram: Functionality



- Also showing unselected features
- Occurence of arbitrary files
- Tooltip containing all methods and fiels of the role
- Tooltip containing all arbitrary files of the same type
- Filter the diagram by a feature or class
- Open the role or class file by doble clicking the box

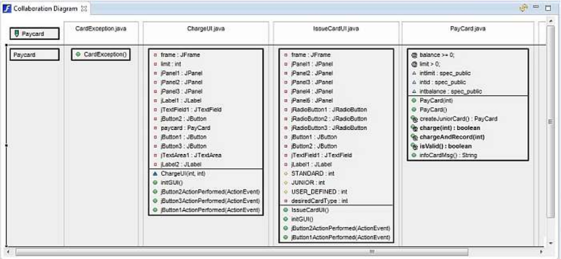
Collaboration Diagram: Functionality



Paycard	CardException.java	ChargeUI.java	IssueCardUI.java	PayCard.java
Paycard	Fields: 0 Methods: 1 Invariants: 0	Fields: 13 Methods: 5 Invariants: 0	Fields: 17 Methods: 4 Invariants: 0	Fields: 3 Methods: 7 Invariants: 2
Logging				Fields: 1 Methods: 1 Invariants: 0
MaximumRecord				
LockOut				Fields: 1 Methods: 2 Invariants: 1

- Showing implementations details: Methods, Fields...

Collaboration Diagram: Functionality



Collaboration Diagram

Paycard	CardException.java	ChargeUI.java	IssueCardUI.java	PayCard.java
Paycard	CardException()	frame : JFrame limit : int jPanel1 : JPanel jPanel2 : JPanel jPanel3 : JPanel jLabel1 : JLabel jTextField1 : JTextField jButton2 : JButton paycard : PayCard jButton1 : JButton jButton3 : JButton jTextArea1 : JTextArea jLabel2 : JLabel ChargeUI(int, int) initGUI() jButton2ActionPerformed(ActionEvent) jButton3ActionPerformed(ActionEvent) jButton1ActionPerformed(ActionEvent)	frame : JFrame jPanel1 : JPanel jPanel2 : JPanel jPanel3 : JPanel jPanel4 : JPanel jPanel6 : JPanel jRadioButton1 : JRadioButton jRadioButton2 : JRadioButton jRadioButton3 : JRadioButton jButton1 : JButton jButton2 : JButton jTextField1 : JTextField jLabel1 : JLabel STANDARD : int JUNIOR : int USER_DEFINED : int desiredCardType : int IssueCardUI() initGUI() jButton2ActionPerformed(ActionEvent) jButton1ActionPerformed(ActionEvent)	balance >= 0; limit > 0; intlimit : spec_public intid : spec_public intbalance : spec_public PayCard(int) PayCard() createJuniorCard() : PayCard **charge(int) : boolean** **chargeAndRecord(int)** **isValid() : boolean** infoCardMsg() : String

- Showing implementations details: Methods, Fields...