OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

FeatureIDE: Development

Thomas Thüm, Jens Meinicke

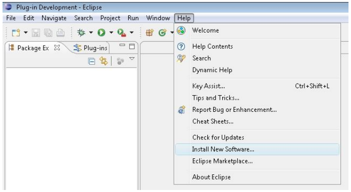
March 4, 2015

Installing Eclipse

1. Download Eclipse: `http://www.eclipse.org/downloads/`
   - 4.3 (Kepler) is recommended (works also with 3.4 and newer)
   - Choose "Eclipse for RCP and RAP Developers" that you can access sources of Eclipse standard plug-ins
2. Unzip Eclipse
   - Make sure that you have all permissions for the directory (do not use Windows' program files folder)
3. Create a shortcut, add VM arguments: `.../eclipse.exe -vmargs -Duser.name="Name Surname" -Xmx1024M`
   - Eclipse can automatically insert your name as author
   - Avoids OutOfMemoryException
4. Start Eclipse and create a new workspace

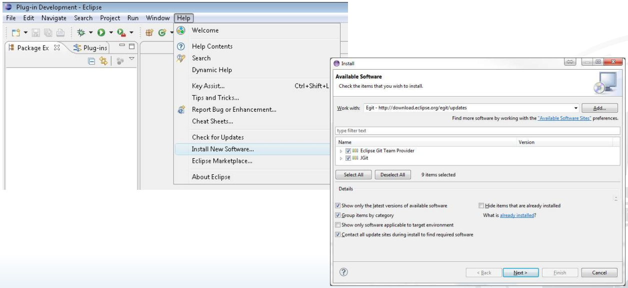
Installing EGit, CDT and FindBugs

5. Install Git plugin such as EGit using Eclipse update mechanism `http://download.eclipse.org/egit/updates`
6. Install CDT (can be skipped if you not intend to work with the FeatureC++ plugin) `http://download.eclipse.org/tools/cdt/releases/8.4`



Installing EGit, CDT and FindBugs

5. Install Git plugin such as EGit using Eclipse update mechanism `http://download.eclipse.org/egit/updates`
6. Install CDT (can be skipped if you not intend to work with the FeatureC++ plugin) `http://download.eclipse.org/tools/cdt/releases/8.4`



Available Software
Check the items that you wish to install.

Work with: Egit - http://download.eclipse.org/egit/updates [Add...]
Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
☑ Eclipse Git Team Provider	
☑ JGit	

[Select All] [Deselect All] 9 items selected

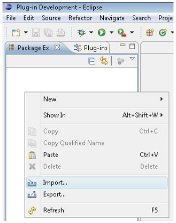
Details

- ☑ Show only the latest versions of available software
- ☑ Group items by category
- ☐ Show only software applicable to target environment
- ☑ Contact all update sites during install to find required software
- ☐ Hide items that are already installed
- What is already installed?

[< Back] [Next >] [Finish] [Cancel]

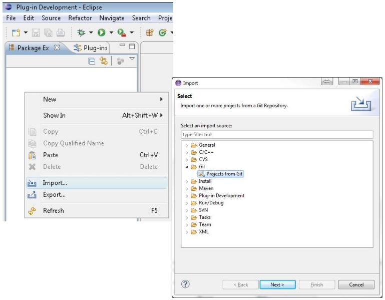
Checkout FeatureIDE Sources

7. Download FeatureIDE plugins from our Git repository
   `https://github.com/tthuem/FeatureIDE.git`
   - no login credentials required for checkout



Checkout FeatureIDE Sources

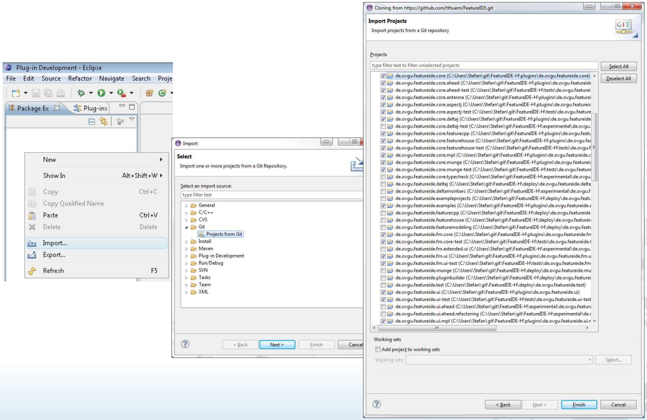
7. Download FeatureIDE plugins from our Git repository
   `https://github.com/tthuem/FeatureIDE.git`
   - no login credentials required for checkout



Import one or more projects from a Git Repository.

Checkout FeatureIDE Sources

7. Download FeatureIDE plugins from our Git repository
   `https://github.com/tthuem/FeatureIDE.git`
   - no login credentials required for checkout



Select

Import one or more projects from a Git Repository.

Select an import source:

type filter text

Cloning from https://github.com/tthuem/FeatureIDE.git

Import Projects

Import projects from a Git repository

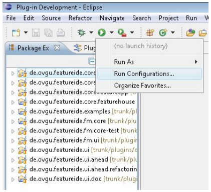
Projects

type filter text to filter unselected projects

```
[✓] de.ovgu.featureide.core (C:\Users\Stefan\git\FeatureIDE-H\plugins\de.ovgu.featureide.core)
[✓] de.ovgu.featureide.core.ahead (C:\Users\Stefan\git\FeatureIDE-H\plugins\de.ovgu.featureide
[✓] de.ovgu.featureide.core.ahead-test (C:\Users\Stefan\git\FeatureIDE-H\tests\de.ovgu.featurei
[✓] de.ovgu.featureide.core.antenna (C:\Users\Stefan\git\FeatureIDE-H\plugins\de.ovgu.featurei
[✓] de.ovgu.featureide.core.aspectj (C:\Users\Stefan\git\FeatureIDE-H\plugins\de.ovgu.featureic
[✓] de.ovgu.featureide.core.aspectj-test (C:\Users\Stefan\git\FeatureIDE-H\tests\de.ovgu.feature
[✓] de.ovgu.featureide.core.deltaj (C:\Users\Stefan\git\FeatureIDE-H\plugins\de.ovgu.featureide
[ ] de.ovgu.featureide.core.deltaj-test (C:\Users\Stefan\git\FeatureIDE-H\experimental\de.ovgu.
[✓] de.ovgu.featureide.core.featurecpp (C:\Users\Stefan\git\FeatureIDE-H\plugins\de.ovgu.featu
[✓] de.ovgu.featureide.core.featurehouse (C:\Users\Stefan\git\FeatureIDE-H\plugins\de.ovgu.fea
[✓] de.ovgu.featureide.core.featurehouse-test (C:\Users\Stefan\git\FeatureIDE-H\tests\de.ovgu.f
[✓] de.ovgu.featureide.core.mpl (C:\Users\Stefan\git\FeatureIDE-H\plugins\de.ovgu.featureide.c
[✓] de.ovgu.featureide.core.munge (C:\Users\Stefan\git\FeatureIDE-H\plugins\de.ovgu.featureic
[✓] de.ovgu.featureide.core.munge-test (C:\Users\Stefan\git\FeatureIDE-H\tests\de.ovgu.feature
[ ] de.ovgu.featureide.core.typecheck (C:\Users\Stefan\git\FeatureIDE-H\experimental\de.ovgu.
[ ] de.ovgu.featureide.deltaj (C:\Users\Stefan\git\FeatureIDE-H\deploy\de.ovgu.featureide.delta
[ ] de.ovgu.featureide.deltamontiarc (C:\Users\Stefan\git\FeatureIDE-H\experimental\de.ovgu.f
[ ] de.ovgu.featureide.exampleprojects (C:\Users\Stefan\git\FeatureIDE-H\deploy\de.ovgu.featu
[✓] de.ovgu.featureide.examples (C:\Users\Stefan\git\FeatureIDE-H\plugins\de.ovgu.featureide.e
[ ] de.ovgu.featureide.featurecpp (C:\Users\Stefan\git\FeatureIDE-H\deploy\de.ovgu.featureide
[ ] de.ovgu.featureide.featurehouse (C:\Users\Stefan\git\FeatureIDE-H\deploy\de.ovgu.featurei
[ ] de.ovgu.featureide.featuremodeling (C:\Users\Stefan\git\FeatureIDE-H\deploy\de.ovgu.featu
[✓] de.ovgu.featureide.fm.core (C:\Users\Stefan\git\FeatureIDE-H\plugins\de.ovgu.featureide.fm
[✓] de.ovgu.featureide.fm.core-test (C:\Users\Stefan\git\FeatureIDE-H\tests\de.ovgu.featureide.f
[ ] de.ovgu.featureide.fm.extended.ui (C:\Users\Stefan\git\FeatureIDE-H\experimental\de.ovgu.
[✓] de.ovgu.featureide.fm.ui (C:\Users\Stefan\git\FeatureIDE-H\plugins\de.ovgu.featureide.fm.u
[✓] de.ovgu.featureide.fm.ui-test (C:\Users\Stefan\git\FeatureIDE-H\tests\de.ovgu.featureide.fm
[ ] de.ovgu.featureide.munge (C:\Users\Stefan\git\FeatureIDE-H\deploy\de.ovgu.featureide.mu
[ ] de.ovgu.featureide.pluginbuilder (C:\Users\Stefan\git\FeatureIDE-H\deploy\de.ovgu.featurei
[ ] de.ovgu.featureide.test (C:\Users\Stefan\git\FeatureIDE-H\deploy\de.ovgu.featureide.test)
[✓] de.ovgu.featureide.ui (C:\Users\Stefan\git\FeatureIDE-H\plugins\de.ovgu.featureide.ui)
[✓] de.ovgu.featureide.ui-test (C:\Users\Stefan\git\FeatureIDE-H\tests\de.ovgu.featureide.ui-test
[ ] de.ovgu.featureide.ui.ahead (C:\Users\Stefan\git\FeatureIDE-H\experimental\de.ovgu.feature
[ ] de.ovgu.featureide.ui.ahead.refactoring (C:\Users\Stefan\git\FeatureIDE-H\experimental\de.c
[✓] de.ovgu.featureide.ui.mpl (C:\Users\Stefan\git\FeatureIDE-H\plugins\de.ovgu.featureide.ui.r
```

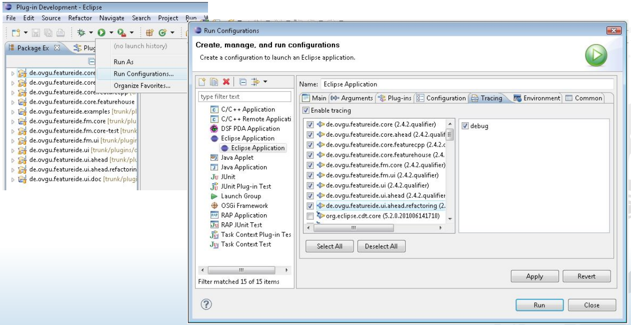
Creating a Run Configuration

8. Create a new run configuration for Eclipse Applications and enable debug tracing for all FeatureIDE plugins (named `de.ovgu.featureide.*`)
9. Also add VM arguments to avoid OutOfMemory Exceptions: `-Dosgi.requiredJavaVersion=1.6 -Xmx512M -XX:MaxPermSize=256M`


Plug-in Development - Eclipse
File
Edit
Source
Refactor
Navigate
Search
Project
Run
Package Ex
Plug
(no launch history)
Run As
Run Configurations...
Organize Favorites...
de.ovgu.featureide.core
de.ovgu.featureide.core
de.ovgu.featureide.core
de.ovgu.featureide.core.featurehouse
de.ovgu.featureide.examples [trunk/pl
de.ovgu.featureide.fm.core [trunk/plu
de.ovgu.featureide.fm.core-test [trunk
de.ovgu.featureide.fm.ui [trunk/plugin
de.ovgu.featureide.ui [trunk/plugins/c
de.ovgu.featureide.ui.ahead [trunk/plu
de.ovgu.featureide.ui.ahead.refactorin
de.ovgu.featureide.ui.doc [trunk/plugi

Creating a Run Configuration

8. Create a new run configuration for Eclipse Applications and enable debug tracing for all FeatureIDE plugins (named `de.ovgu.featureide.*`)
9. Also add VM arguments to avoid OutOfMemory Exceptions: `-Dosgi.requiredJavaVersion=1.6 -Xmx512M -XX:MaxPermSize=256M`



de.ovgu.featureide.core
de.ovgu.featureide.core
de.ovgu.featureide.core
de.ovgu.featureide.core.featurehouse
de.ovgu.featureide.examples [trunk/pl
de.ovgu.featureide.fm.core [trunk/plu
de.ovgu.featureide.fm.core-test [trunk
de.ovgu.featureide.fm.ui [trunk/plugin
de.ovgu.featureide.ui [trunk/plugins/c
de.ovgu.featureide.ui.ahead [trunk/plu
de.ovgu.featureide.ui.ahead.refactorin
de.ovgu.featureide.ui.doc [trunk/plugi

Name: Eclipse Application

☑ Enable tracing

☑ de.ovgu.featureide.core (2.4.2.qualifier)
☑ de.ovgu.featureide.core.ahead (2.4.2.qualif
☑ de.ovgu.featureide.core.featurecpp (2.4.2.c
☑ de.ovgu.featureide.core.featurehouse (2.4.
☑ de.ovgu.featureide.fm.core (2.4.2.qualifier)
☑ de.ovgu.featureide.fm.ui (2.4.2.qualifier)
☑ de.ovgu.featureide.ui (2.4.2.qualifier)
☑ de.ovgu.featureide.ui.ahead (2.4.2.qualifier
☑ de.ovgu.featureide.ui.ahead.refactoring (2.
☐ org.eclipse.cdt.core (5.2.0.201006141710)

☑ debug

Structure of the Repository

`plugins/`	Source Code of the FeatureIDE plugins
`deploy/`	FeatureIDE features, update site project, and plugin builder project
`lib/`	Extensions of AHEAD used in FeatureIDE
`featuremodels/`	Example FeatureIDE projects, feature model without code
`tests/`	JUnit test plugins
`experimental/`	Non-stable implementations

FeatureIDE Features

1. Feature Modeling:
   `de.ovgu.featureide.featuremodeling`
2. FeatureIDE (requires 1.):
   `de.ovgu.featureide`
3. FeatureIDE extension for FeatureHouse (requires 2.):
   `de.ovgu.featureide.featurehouse`
4. FeatureIDE extension for FeatureC++ (requires 2.):
   `de.ovgu.featureide.featurecpp`
5. FeatureIDE extension for Antenna (requires 2.):
   `de.ovgu.featureide.antenna`

FeatureIDE Features

6. FeatureIDE extension for AspectJ (requires 2.):
   `de.ovgu.featureide.aspectj`
7. FeatureIDE extension for DeltaJ (requires 2.):
   `de.ovgu.featureide.deltaj`
8. FeatureIDE extension for Munge (requires 2.):
   `de.ovgu.featureide.munge`
9. Unit-Tests for FeatureIDE
10. FeatureIDE example projects

1. Feature Modeling

Plugins in Feature `de.ovgu.featureide.featuremodeling`:

- `de.ovgu.featureide.fm.core`
  - Abstract models for feature models and configurations
  - Parser and writer for feature models and configurations
  - Automated analysis for feature models and configurations
  - Classification of feature model edits
- `de.ovgu.featureide.fm.ui`
  - Feature Model Editor
  - Error markers for feature models and configurations
  - Feature Model Edit View
  - Feature Model Outline View
  - Import, export, and printing of feature models

2. FeatureIDE

Core plugins in Feature `de.ovgu.featureide`:

- `de.ovgu.featureide.core`
  - Abstract feature project
  - Extensible builder
  - Abstract FSTModel
- `de.ovgu.featureide.core.ahead`
  - Builder extension to compose Jak files
  - Full FSTModel for Jak files
  - Localization of Jak errors in source files

2. FeatureIDE

UI plugins in Feature `de.ovgu.featureide`:

- `de.ovgu.featureide.ui`
  - FeatureIDE perspective
  - Decorators, buttons, and menu items
  - Collaboration Diagram
  - Collaboration Outline View
  - Feature Statistics View
  - Wizards for FeatureIDE projects, configurations and files
  - Builder to create all valid or current products
- `de.ovgu.featureide.ui.ahead`
  - Jak editor with content assist and outline view
- `de.ovgu.featureide.ui.doc`
  - Cheat sheet and FeatureIDE introduction page

3./4. FeatureHouse and FeatureC++ Extension

Plugins in Feature `de.ovgu.featureide.featurehouse`:

- `de.ovgu.featureide.core.featurehouse`
  - Builder extension to compose FeatureHouse files
  - Error Propagation for Feature House files
  - FSTModel for the actual Configuration
  - Support for contracts in JML

Plugins in Feature `de.ovgu.featureide.featurecpp`:

- `de.ovgu.featureide.core.featurecpp`
  - Builder extension to compose FeatureC++ files
  - FSTModel for the actual Configuration

5./6. Antenna and AspectJ Extension

Plugins in Feature `de.ovgu.featureide.antenna`:

- `de.ovgu.featureide.core.antenna`
    - Preprocessor extension using Antenna
    - FSTModel with preprocessor annotations

Plugins in Feature `de.ovgu.featureide.aspectj`:

- `de.ovgu.featureide.core.aspectj`
    - Builder extension using AspectJ

7./8. DeltaJ and Munge Extension

Plugins in Feature `de.ovgu.featureide.deltaj`:

- `de.ovgu.featureide.core.deltaj`
  - Builder extension using DeltaJ

Plugins in Feature `de.ovgu.featureide.munge`:

- `de.ovgu.featureide.core.munge`
  - Preprocessor extension using Munge
  - FSTModel with preprocessor annotations
  - Error propagation

9./10. JUnit Tests and Examples

Plugins in Feature `de.ovgu.featureide.test`:

- `de.ovgu.featureide.*-test`
  - Several JUnit tests

Plugins in Feature `de.ovgu.featureide.examples`:

- `de.ovgu.featureide.examples`
  - Example projects for several composition tools

Extension Points

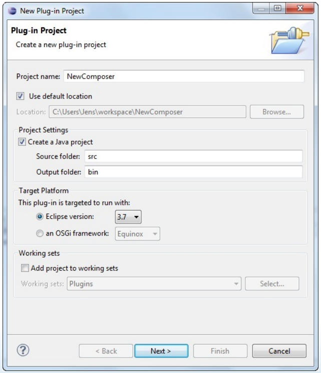
You can extend FeatureIDE with your own functionality, by using the provided extension points named:

1. `de.ovgu.featureide.core.composers`
2. `de.ovgu.featureide.fm.core.FMComposer`
3. `de.ovgu.featureide.fm.ui.FeatureDiagram`
4. `de.ovgu.featureide.fm.ui.FeatureModelEditor`
5. `de.ovgu.featureide.ui.ConfigurationEditor`

Integrate a Composition Tool

Create a new Plug-in Project (open the Plug-in Project wizard)

- Set the projects name
- Disable activator generation
- Press Finish



New Plug-in Project

Plug-in Project

Create a new plug-in project

Project name: NewComposer

[✓] Use default location

Location: C:\Users\Jens\workspace\NewComposer

Project Settings

[✓] Create a Java project

Source folder: src

Output folder: bin

Target Platform

This plug-in is targeted to run with:

[● Eclipse version: 3.7]

○ an OSGi framework: Equinox

Working sets

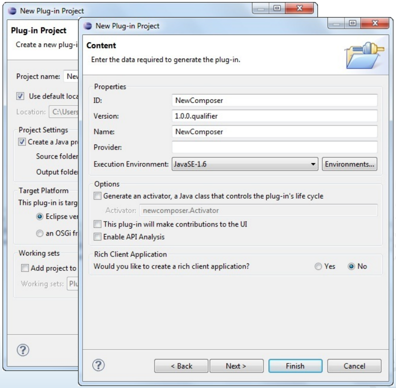
[ ] Add project to working sets

Working sets: Plugins

Integrate a Composition Tool

Create a new Plug-in Project (open the Plug-in Project wizard)

- Set the projects name
- Disable activator generation
- Press Finish



New Plug-in Project

Plug-in Project
Create a new plug-i

Project name: Nev
[✓] Use default loca
Location: C:\Users

Project Settings
[✓] Create a Java pr
Source folder
Output folde

Target Platform
This plug-in is targ
[✓ Eclipse ver]
an OSGi fr

Working sets
[ ] Add project to
Working sets: Plu

New Plug-in Project

Content
Enter the data required to generate the plug-in.

Properties

ID:	NewComposer
Version:	1.0.0.qualifier
Name:	NewComposer
Provider:	
Execution Environment:	JavaSE-1.6

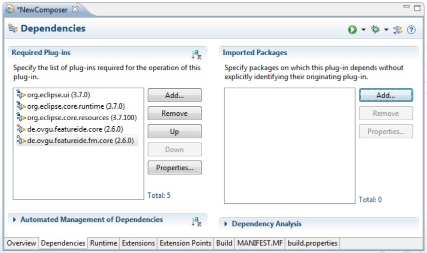
Options
[ ] Generate an activator, a Java class that controls the plug-in's life cycle
Activator: newcomposer.Activator
[ ] This plug-in will make contributions to the UI
[ ] Enable API Analysis

Rich Client Application
Would you like to create a rich client application? Yes [✓ No]

Setup the Plug-in Project

After creating the project, the plugin manifest will be opened

- Select the Dependencies page and add following required plug-ins
  - org.eclipse.ui
  - org.eclipse.core.runtime
  - org.eclipse.core.resources
  - de.ovgu.featureide.core
  - de.ovgu.featureide.fm.core


*NewComposer

Dependencies

Required Plug-ins

Specify the list of plug-ins required for the operation of this plug-in.

org.eclipse.ui (3.7.0)
org.eclipse.core.runtime (3.7.0)
org.eclipse.core.resources (3.7.100)
de.ovgu.featureide.core (2.6.0)
de.ovgu.featureide.fm.core (2.6.0)

Total: 5

Imported Packages

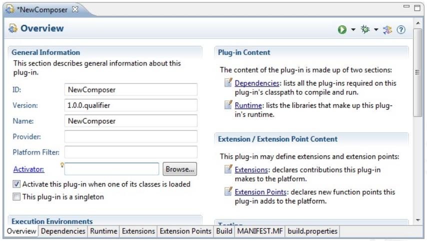
Specify packages on which this plug-in depends without explicitly identifying their originating plug-in.

Total: 0

Setup the Plug-in Project

Select the Overview Page

- enable plug-in-activation when a class is loaded
- left-click “Activator:“ to create the activator class


*NewComposer

Overview

General Information

This section describes general information about this plug-in.

ID: NewComposer
Version: 1.0.0.qualifier
Name: NewComposer
Provider:
Platform Filter:
Activator:

[✓] Activate this plug-in when one of its classes is loaded
[ ] This plug-in is a singleton

Execution Environments

Plug-in Content

The content of the plug-in is made up of two sections:

Dependencies: lists all the plug-ins required on this plug-in's classpath to compile and run.

Runtime: lists the libraries that make up this plug-in's runtime.

Extension / Extension Point Content

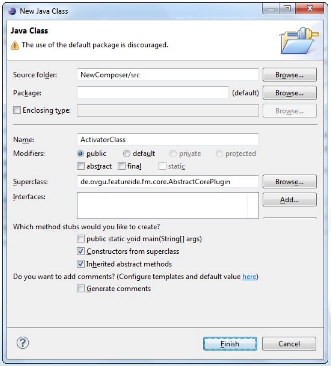
This plug-in may define extensions and extension points:

Extensions: declares contributions this plug-in makes to the platform.

Extension Points: declares new function points this plug-in adds to the platform.

Create the Activator Class

- Set the Activators name
- Set the package (not deafault package)
- Set the Superclass:
  - de.ovgu.featureide.fm.core.AbstractCorePlugin
- Press Finish


New Java Class

Java Class

The use of the default package is discouraged.

Source folder:	NewComposer/src	Browse...
Package:	(default)	Browse...
☐ Enclosing type:		Browse...
Name:	ActivatorClass	
Modifiers:	● public ○ default ○ private ○ protected	
	☐ abstract ☐ final ☐ static	
Superclass:	de.ovgu.featureide.fm.core.AbstractCorePlugin	Browse...
Interfaces:		Add...

Which method stubs would you like to create?

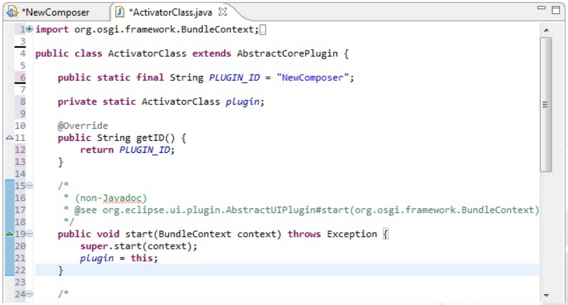
☐ public static void main(String[] args)
☑ Constructors from superclass
☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

Create the Activator Class

The new activator class should look like the activator classes of the other composer plug-ins. e.g. look at AheadCorePlugin of the AHEAD plug-in.



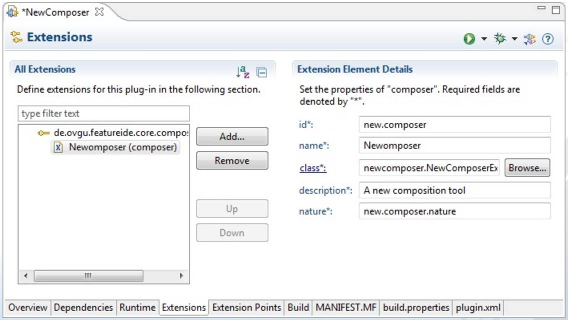
*NewComposer | *ActivatorClass.java

```
import org.osgi.framework.BundleContext;

public class ActivatorClass extends AbstractCorePlugin {

    public static final String PLUGIN_ID = "NewComposer";

    private static ActivatorClass plugin;

    @Override
    public String getID() {
        return PLUGIN_ID;
    }

    /*
     * (non-Javadoc)
     * @see org.eclipse.ui.plugin.AbstractUIPlugin#start(org.osgi.framework.BundleContext)
     */
    public void start(BundleContext context) throws Exception {
        super.start(context);
        plugin = this;
    }

    /*
```

Set Extension Point

Select the Extensions Page

- add the following extension point
  - de.ovgu.featureide.core.composers
- Specify the Extension Element Details on the right
- left-click “class*:“ to create the composer class
  - the file wizard is auto-filled, so only press finish



*NewComposer

Extensions

All Extensions

Define extensions for this plug-in in the following section.

type filter text

de.ovgu.featureide.core.compo:
Newomposer (composer)

Extension Element Details

Set the properties of "composer". Required fields are denoted by "*".

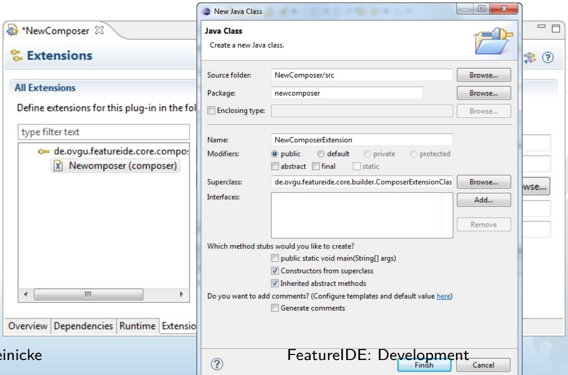
id*:	new.composer
name*:	Newomposer
class*:	newcomposer.NewComposerEx
description*:	A new composition tool
nature*:	new.composer.nature

Browse...
Overview | Dependencies | Runtime | Extensions | Extension Points | Build | MANIFEST.MF | build.properties | plugin.xml

Set Extension Point

Select the Extensions Page

- add the following extension point
  - de.ovgu.featureide.core.composers
- Specify the Extension Element Details on the right
- left-click “class*:“ to create the composer class
  - the file wizard is auto-filled, so only press finish



*NewComposer

Extensions

All Extensions

Define extensions for this plug-in in the fol

type filter text

de.ovgu.featureide.core.compo:

Newomposer (composer)

Overview | Dependencies | Runtime | Extensio

New Java Class

Java Class

Create a new Java class.

Source folder: NewComposer/src

Package: newcomposer

[ ] Enclosing type:

Name: NewComposerExtension

Modifiers: [✓ public] default private protected

[ ] abstract [ ] final [ ] static

Superclass: de.ovgu.featureide.core.builder.ComposerExtensionClas

Interfaces:

Which method stubs would you like to create?

[ ] public static void main(String[] args)

[✓] Constructors from superclass

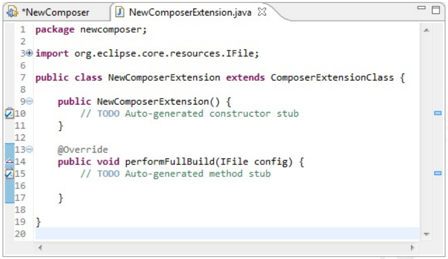
[✓] Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

[ ] Generate comments

Composer Integration

To integrate your composition tool you need to implement the newly created class. The most methods got default implementations. To adjust FeatureIDE to your composer, implement the provided methods. For further informations see their Javadoc.



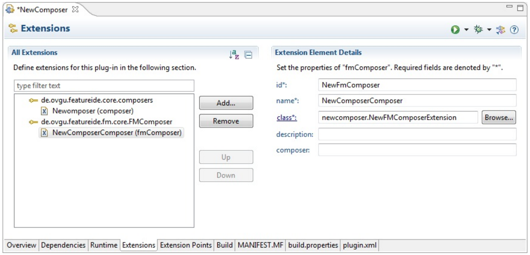
```java
package newcomposer;

import org.eclipse.core.resources.IFile;

public class NewComposerExtension extends ComposerExtensionClass {

    public NewComposerExtension() {
        // TODO Auto-generated constructor stub
    }

    @Override
    public void performFullBuild(IFile config) {
        // TODO Auto-generated method stub

    }

}
```

FMComposer Extension

For some Feature Model specific extensions (e.g. renamings and feature order.) you need to extend the following extension point.

- de.ovgu.featureide.fm.core.FMComposer

Create and implement the class of this extension point



Define extensions for this plug-in in the following section.

type filter text

- de.ovgu.featureide.core.composers
  - Newomposer (composer)
- de.ovgu.featureide.fm.core.FMComposer
  - NewComposerComposer (fmComposer)

Extension Element Details

Set the properties of "fmComposer". Required fields are denoted by "*".

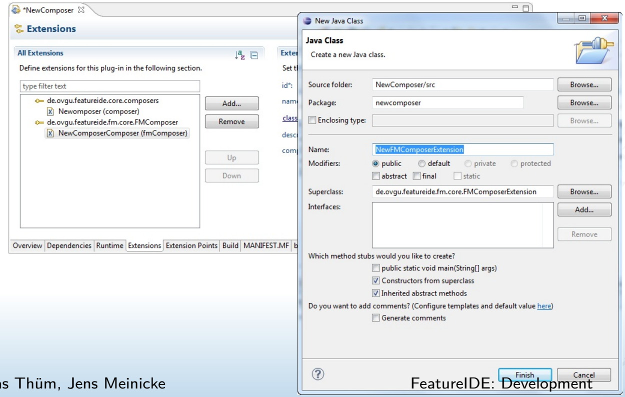
id*:	NewFmComposer
name*:	NewComposerComposer
class*:	newcomposer.NewFMComposerExtension
description:	
composer:	

FMComposer Extension

For some Feature Model specific extensions (e.g. renamings and feature order.) you need to extend the following extension point.

- de.ovgu.featureide.fm.core.FMComposer

Create and implement the class of this extension point



*NewComposer

Extensions

All Extensions

Define extensions for this plug-in in the following section.

type filter text

- de.ovgu.featureide.core.composers
  - Newomposer (composer)
- de.ovgu.featureide.fm.core.FMComposer
  - NewComposerComposer (fmComposer)

Add...
Remove
Up
Down
Overview | Dependencies | Runtime | Extensions | Extension Points | Build | MANIFEST.MF

New Java Class

Java Class

Create a new Java class.

Source folder: NewComposer/src

Package: newcomposer

Enclosing type:

Name: NewFMComposerExtension

Modifiers: [✓ public] default private protected

abstract final static

Superclass: de.ovgu.featureide.fm.core.FMComposerExtension

Interfaces:

Which method stubs would you like to create?

- [ ] public static void main(String[] args)
- [x] Constructors from superclass
- [x] Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

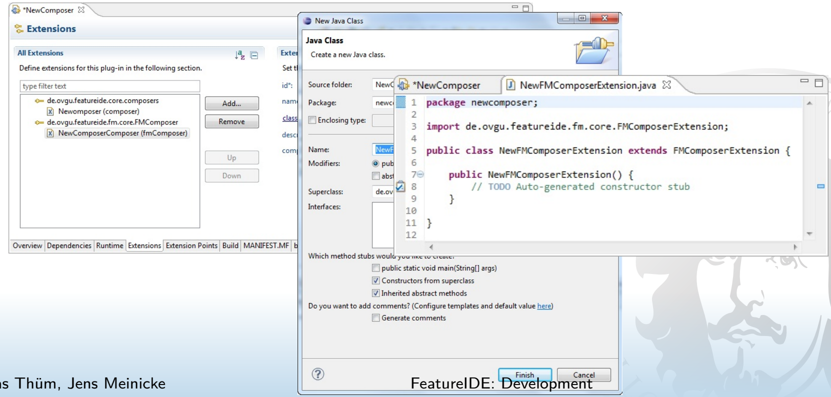
- [ ] Generate comments

FMComposer Extension

For some Feature Model specific extensions (e.g. renamings and feature order.) you need to extend the following extension point.

- de.ovgu.featureide.fm.core.FMComposer

Create and implement the class of this extension point

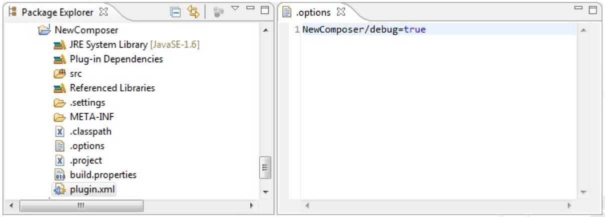

*NewComposer
Extensions
All Extensions
Define extensions for this plug-in in the following section.
type filter text
de.ovgu.featureide.core.composers
Newomposer (composer)
de.ovgu.featureide.fm.core.FMComposer
NewComposerComposer (fmComposer)
Add...
Remove
Up
Down
Overview
Dependencies
Runtime
Extensions
Extension Points
Build
MANIFEST.MF
New Java Class
Java Class
Create a new Java class.
Source folder:
Package:
Enclosing type:
Name:
Modifiers:
Superclass:
Interfaces:
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel
NewFMComposerExtension.java

```java
package newcomposer;

import de.ovgu.featureide.fm.core.FMComposerExtension;

public class NewFMComposerExtension extends FMComposerExtension {

	public NewFMComposerExtension() {
		// TODO Auto-generated constructor stub
	}

}
```

Debugging

To get debugging information you need to create a file named ".options" at your plug-in project. Set the files content to "plug-in-id"/debug=true Now you can do outputs to the errorlog by calling:

- ActivatorClass.getDefault().log...();

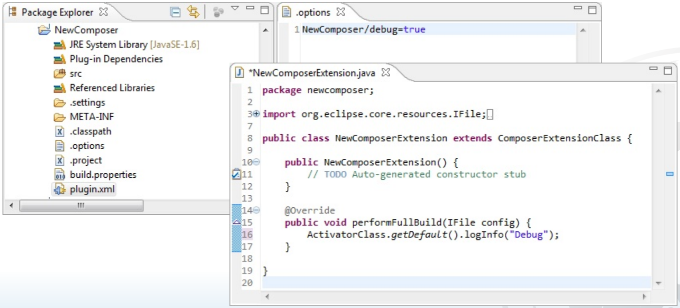


```
NewComposer/debug=true
```

Debugging

To get debugging information you need to create a file named ".options" at your plug-in project. Set the files content to "plug-in-id"/debug=true Now you can do outputs to the errorlog by calling:

- ActivatorClass.getDefault().log...();



```
NewComposer/debug=true
```

```
package newcomposer;

import org.eclipse.core.resources.IFile;

public class NewComposerExtension extends ComposerExtensionClass {

    public NewComposerExtension() {
        // TODO Auto-generated constructor stub
    }

    @Override
    public void performFullBuild(IFile config) {
        ActivatorClass.getDefault().logInfo("Debug");
    }

}
```

Debugging with FindBugs

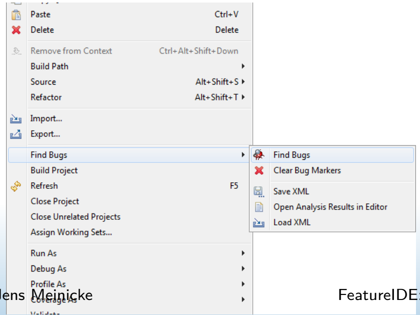
FindBugs is a static analyzation tool for Java. It shows errors comparable to the Java compiler but it is much more powerful.

- Install Findbugs via updatesite: `http://findbugs.cs.umd.edu/eclipse`

See also: `http://findbugs.sourceforge.net/`

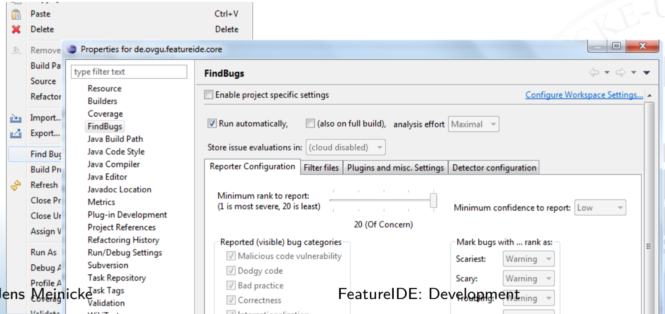
Run FindBugs

- Run manual
  - Open the context menu of a java project
  - Open the submenu of 'Find Bugs'
  - Run 'Find Bugs'
- Run automatically
  - Open the property page of a Java Project.
  - Open the entry for FindBugs.
  - Activate 'Run Automatically'



Run FindBugs

- Run manual
  - Open the context menu of a java project
  - Open the submenu of 'Find Bugs'
  - Run 'Find Bugs'
- Run automatically
  - Open the property page of a Java Project.
  - Open the entry for FindBugs.
  - Activate 'Run Automatically'



Properties for de.ovgu.featureide.core

FindBugs

☐ Enable project specific settings

☑ Run automatically, ☐ (also on full build), analysis effort Maximal

Store issue evaluations in: (cloud disabled)

Minimum rank to report:
(1 is most severe, 20 is least)
20 (Of Concern)

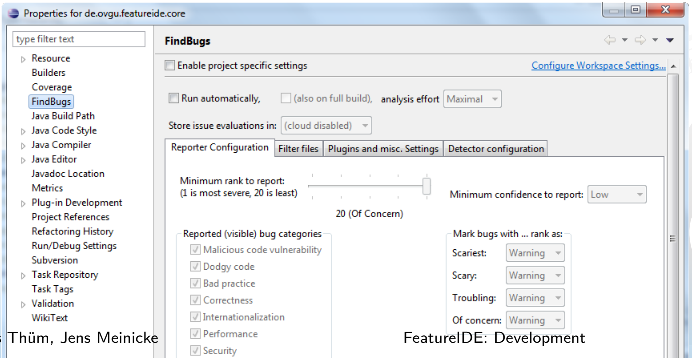
Minimum confidence to report: Low

Reported (visible) bug categories
☑ Malicious code vulnerability
☑ Dodgy code
☑ Bad practice
☑ Correctness

Mark bugs with ... rank as:
Scariest: Warning
Scary: Warning
Troubling: Warning

Configure FindBugs

- Open the property page of a Java Project.
- Open the entry for FindBugs.
- Select 'Configure Workspace Settings' at the upper right corner.
- Here you can specify settings for the whole workspce.


Properties for de.ovgu.featureide.core

FindBugs

Enable project specific settings — Configure Workspace Settings...

Run automatically, (also on full build), analysis effort: Maximal

Store issue evaluations in: (cloud disabled)

Minimum rank to report: (1 is most severe, 20 is least) — 20 (Of Concern)

Minimum confidence to report: Low

Reported (visible) bug categories

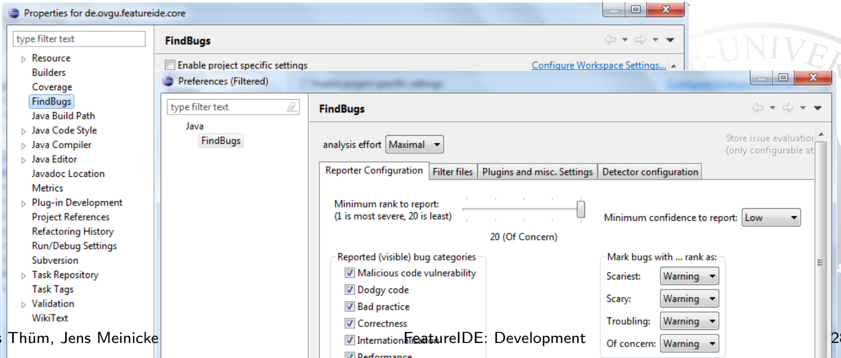
- ✓ Malicious code vulnerability
- ✓ Dodgy code
- ✓ Bad practice
- ✓ Correctness
- ✓ Internationalization
- ✓ Performance
- ✓ Security

Mark bugs with ... rank as:

Scariest:	Warning
Scary:	Warning
Troubling:	Warning
Of concern:	Warning

Configure FindBugs

- Open the property page of a Java Project.
- Open the entry for FindBugs.
- Select 'Configure Workspace Settings' at the upper right corner.
- Here you can specify settings for the whole workspce.



Properties for de.ovgu.featureide.core

type filter text

Resource
Builders
Coverage
FindBugs
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Metrics
Plug-in Development
Project References
Refactoring History
Run/Debug Settings
Subversion
Task Repository
Task Tags
Validation
WikiText

FindBugs

Enable project specific settings

Configure Workspace Settings...

Preferences (Filtered)

type filter text

Java
FindBugs

FindBugs

analysis effort: Maximal

Store issue evaluation (only configurable at

Reporter Configuration | Filter files | Plugins and misc. Settings | Detector configuration

Minimum rank to report:
(1 is most severe, 20 is least)

20 (Of Concern)

Minimum confidence to report: Low

Reported (visible) bug categories
- [✓] Malicious code vulnerability
- [✓] Dodgy code
- [✓] Bad practice
- [✓] Correctness
- [✓] Internationalization
- [✓] Performance

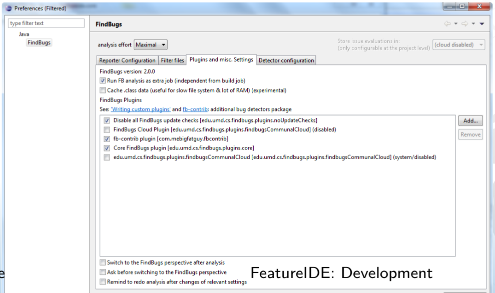
Mark bugs with ... rank as:

Scariest:	Warning
Scary:	Warning
Troubling:	Warning
Of concern:	Warning

Configure FindBugs

To use FindBugs more efficient you need to integrate a jar file named fb-contrib which provides some additional bug patterns.

- Select the tab 'Plugins and misc. Settings'.
- The link will lead you to the website where you can download the jar file.
- Copy the fb-contrib jar into '../eclipse/plugins/edu.umd../plugin/'
- Activate 'fb-contib plugin'



FindBugs

analysis effort: Maximal

Store issue evaluations in: (only configurable at the project level) (cloud disabled)

FindBugs version: 2.0.0

- [✓] Run FB analysis as extra job (independent from build job)
- [ ] Cache .class data (useful for slow file system & lot of RAM) (experimental)

FindBugs Plugins

See: 'Writing custom plugins' and fb-contrib: additional bug detectors package

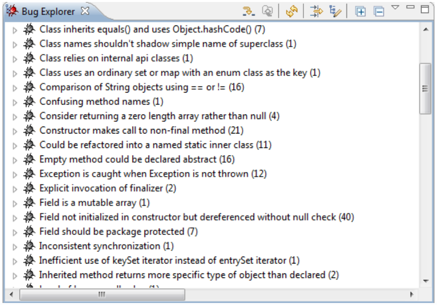
- [✓] Disable all FindBugs update checks [edu.umd.cs.findbugs.plugins.noUpdateChecks]
- [ ] FindBugs Cloud Plugin [edu.umd.cs.findbugs.plugins.findbugsCommunalCloud] (disabled)
- [✓] fb-contrib plugin [com.mebigfatguy.fbcontrib]
- [✓] Core FindBugs plugin [edu.umd.cs.findbugs.plugins.core]
- [ ] edu.umd.cs.findbugs.plugins.findbugsCommunalCloud [edu.umd.cs.findbugs.plugins.findbugsCommunalCloud] (system/disabled)

- [ ] Switch to the FindBugs perspective after analysis
- [ ] Ask before switching to the FindBugs perspective
- [ ] Remind to redo analysis after changes of relevant settings

FindBugs-Views

FindBugs provided some additional views

- Bug Explorer
  - An additional view comparable to the Problems view
- Bug Info
  - Shows additional information about the bug

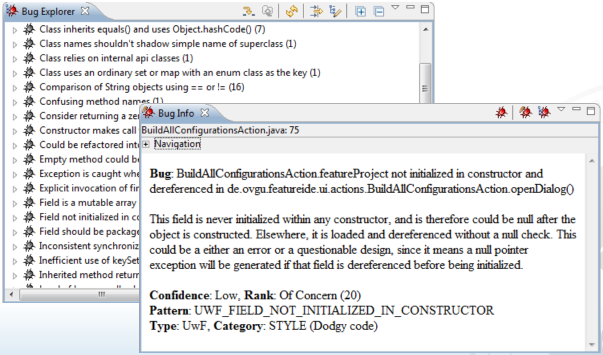


- Class inherits equals() and uses Object.hashCode() (7)
- Class names shouldn't shadow simple name of superclass (1)
- Class relies on internal api classes (1)
- Class uses an ordinary set or map with an enum class as the key (1)
- Comparison of String objects using == or != (16)
- Confusing method names (1)
- Consider returning a zero length array rather than null (4)
- Constructor makes call to non-final method (21)
- Could be refactored into a named static inner class (11)
- Empty method could be declared abstract (16)
- Exception is caught when Exception is not thrown (12)
- Explicit invocation of finalizer (2)
- Field is a mutable array (1)
- Field not initialized in constructor but dereferenced without null check (40)
- Field should be package protected (7)
- Inconsistent synchronization (1)
- Inefficient use of keySet iterator instead of entrySet iterator (1)
- Inherited method returns more specific type of object than declared (2)

FindBugs-Views

FindBugs provided some additional views

- Bug Explorer
  - An additional view comparable to the Problems view
- Bug Info
  - Shows additional information about the bug



Class inherits equals() and uses Object.hashCode() (7)
Class names shouldn't shadow simple name of superclass (1)
Class relies on internal api classes (1)
Class uses an ordinary set or map with an enum class as the key (1)
Comparison of String objects using == or != (16)
Confusing method names (1)
Consider returning a ze
Constructor makes call
Could be refactored int
Empty method could b
Exception is caught whe
Explicit invocation of fi
Field is a mutable array
Field not initialized in c
Field should be package
Inconsistent synchroniz
Inefficient use of keySet
Inherited method retur

BuildAllConfigurationsAction.java: 75

Bug: BuildAllConfigurationsAction.featureProject not initialized in constructor and dereferenced in de.ovgu.featureide.ui.actions.BuildAllConfigurationsAction.openDialog()

This field is never initialized within any constructor, and is therefore could be null after the object is constructed. Elsewhere, it is loaded and dereferenced without a null check. This could be a either an error or a questionable design, since it means a null pointer exception will be generated if that field is dereferenced before being initialized.

Confidence: Low, **Rank**: Of Concern (20)
Pattern: UWF_FIELD_NOT_INITIALIZED_IN_CONSTRUCTOR
Type: UwF, **Category**: STYLE (Dodgy code)

FindBugs-@Annotations

To find some more bugs, FindBugs provides some annotations. To use these annotations add the corresponding jar files to the build path

- jsr305.jar

You will find them at the '*.fm.core plugin' or at the FindBugs installation folder/lib.

FindBugs-@Annotations

Specify the return value or a parameter of a method with an annotation e.g.:

- @Nonnull
  - The value will never be null
  - You need to be sure that the return value is never null or your method does not support null as an argument
  - No unnecessary null checks
- @CheckForNull
  - The value can be null and should be checked

These annotations are important because the most errors are NullPointer and they can be prevented with these annotations. Examples:

- @CheckForNull Object canBeNull(){}
- void checkParameter(@CheckForNull Object parameter){}

Extension Point Architecture

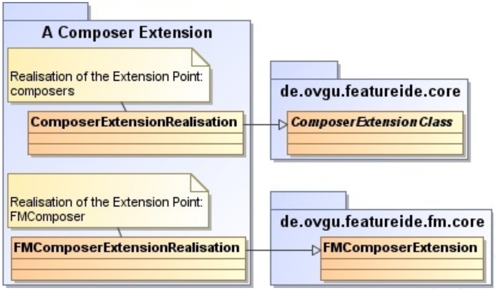

A Composer Extension
Realisation of the Extension Point: composers
ComposerExtensionRealisation
Realisation of the Extension Point: FMComposer
FMComposerExtensionRealisation
de.ovgu.featureide.core
ComposerExtensionClass
de.ovgu.featureide.fm.core
FMComposerExtension

Figure: A simplyfied model of the composer extension points

Preprocessor Integration

To integrate a Preprocessor into FeatureIDE there are Classes with special functionality for Preprocessors

- PPComposerExtensionClass
  - Abstract class to integrate the preprocessor
  - ComposerExtensionClass with predefined functions
- PPModelBuilder
  - Builds a special FSTModel for Preporcessors
  - Shows the occurence of a feature in a class
  - Adds preprocessor directives to the model

Preprocessor Integration

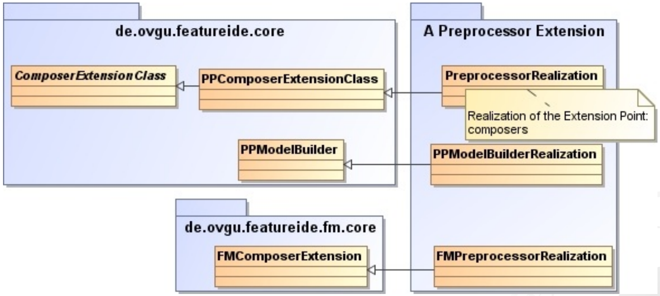


Figure: A simplyfied model of Preprocessor Integration

Feature Model Editor Extension

To extend the feature model editor you need to use the Extension Point:

- de.ovgu.featureide.fm.ui.FeatureModelEditor

With this extension you can

- add new pages to the feature model editor

Feature Model Editor Extension

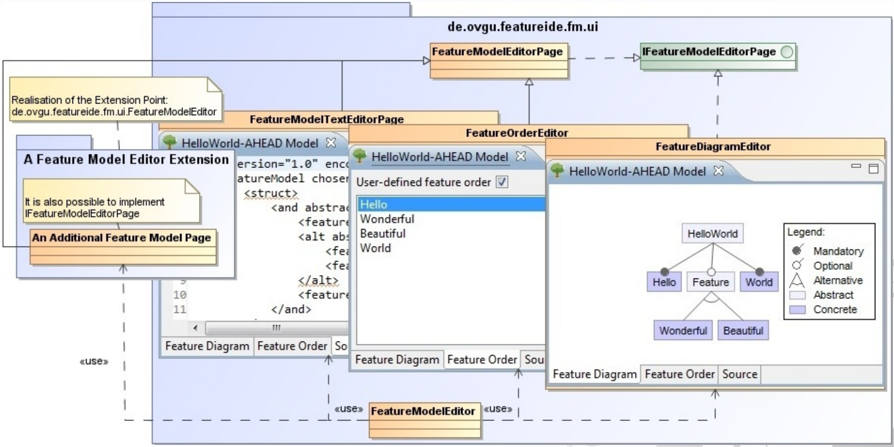


Figure: A simplyfied model of the Feature Model Editor extension point

Feature Diagram Extension

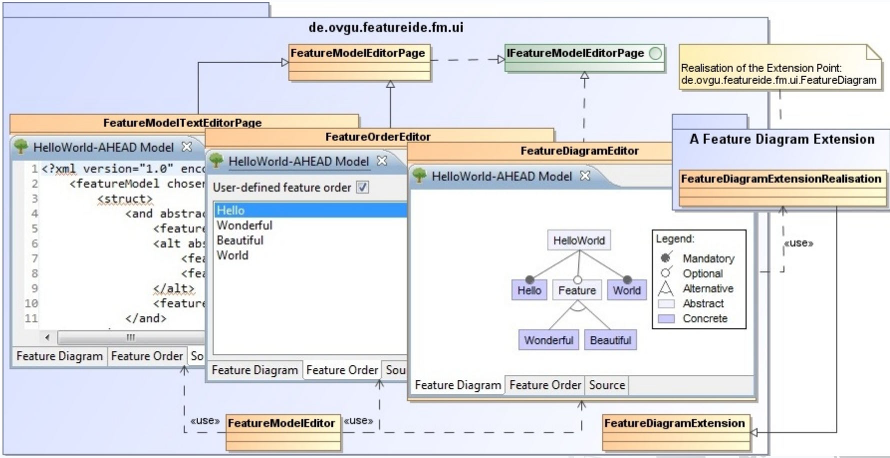
To extend the feature diagram you need to use the Extension Point:

- de.ovgu.featureide.fm.ui.FeatureDiagram

With this extension you can

- extend tooltips
- extend the context menu

Feature Diagram Extension



```
<?xml version="1.0" enco
    <featureModel choser
        <struct>
            <and abstrac
                <feature
                <alt abs
                    <fea
                    <fea
                </alt>
                <feature
            </and>
```

Figure: A simplyfied model of the Feature Diagram extension point

Configuration Editor Extension

To extend the configuration editor you need to use the Extension Point:

- de.ovgu.featureide.ui.ConfigurationEditor

With this extension you can

- add new pages to the configuration editor

Configuration Editor Extension

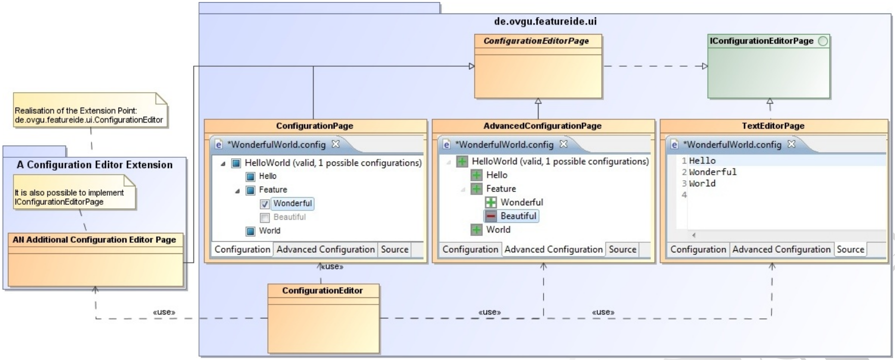

de.ovgu.featureide.ui
ConfigurationEditorPage
IConfigurationEditorPage
Realisation of the Extension Point: de.ovgu.featureide.ui.ConfigurationEditor
A Configuration Editor Extension
It is also possible to implement IConfigurationEditorPage
AN Additional Configuration Editor Page
ConfigurationPage
AdvancedConfigurationPage
TextEditorPage
*WonderfulWorld.config
HelloWorld (valid, 1 possible configurations)
Hello
Feature
Wonderful
Beautiful
World
Configuration
Advanced Configuration
Source
1 Hello
2 Wonderful
3 World
ConfigurationEditor
«use»
«use»
«use»
«use»

Figure: A simplyfied model of the Configuration Editor extension point