FeatureIDE Plug-in URL Association Extension User Manual

Mariana Bonito

m.bonito@campus.fct.unl.pt

Advisors: João Araujo, Armanda Rodrigues
Faculdade de Ciências e Tecnologia – Universidade NOVA de Lisboa

2020

Contents

1 Introduction

This manual focuses on the FeatureIDE plug-in extension which associates URLs to individual features. It covers and explains how to download and run the plug-in code, along with how to use this new component, and all the prerequisites to execute it.

FeatureIDE is a plug-in for the Eclipse software which allows the user to create feature models used in feature-oriented software development. For more information on the FeatureIDE plug-in please see http://www.featureide.com/[5].

This plug-in extension was developed as part of a student investigation and section 5 explains how to visualize the feature models developed for the author's work.

2 Prerequisites

Important: The extension code was developed for the Eclipse version 4.15.0 (2020-03) and based on the FeatureIDE version 3.6.3. It will likely run in updated versions but is untested and cannot be known for sure.

2.1 Oracle JDK

In order to install Eclipse and later run the code, the user must have at least the JDK 8 version. If this is not the case, please follow the instructions on https://docs.oracle.com/en/java/javase/11/install/index.html under the user's operating system which can be found on the table of contents on the left [6].

Please be aware that in some cases when installing on Microsoft Windows, the user might be asked to create an Oracle account. If the user is willing, do so as the installation is simpler. However, if the user does not wish to create an Oracle account, follow the instruction to install the JDK silently https://docs.oracle.com/en/java/javase/11/install/installation-jdk-microsoft-windows-platforms.html#GUID-E3C75F92-D3B2-421D-A9BE-933C15F7CD1B.

2.2 Eclipse IDE

In order to download and run the plug-in extension code, first, the user must have the Eclipse IDE for Java Developers software. If this is not the case please install the software from https://www.eclipse.org/downloads/packages/installer[2] and follow the indicated steps.

Note: In step 3, the package required is the *Eclipse IDE for Java Developers*.

Once the Eclipse application has been opened it should display a welcome tab similar to the one displayed in figure 5. This is there to help the user explore Eclipse, but no interaction with it is necessary.

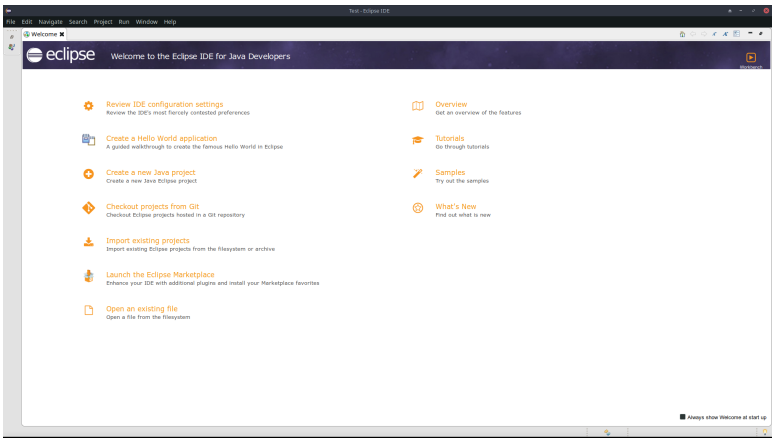

Figure 1: Eclipse welcome page

2.3 FeatureIDE Plug-in for Eclipse IDE

This plug-in is suggested to be installed so that all necessary dependencies are already installed. It avoids unnecessary complications when the user tries to run the code. However, this plug-in is not necessary. The user may wish to try and run the code without this plug-in and only install the dependencies Eclipse indicates are missing. However, this approach is not covered in the manual and the user is fully responsible to figure out the problems associated with it.

To install the FeatureIDE Plug-in the user must go to the *Help* tab on the top of the Eclipse software, select *Eclipse Marketplace*, search *featureIDE*, and press install. Figure 2 is the current FeatureIDE logo to help the user identify the desired plug-in. Now, follow the requested steps and accept terms of use.

Figure 2: FeatureIDE Logo [3]

When the pop-up requests the user to *Confirm Selected Features* as displayed in figure 3, the user can select whichever options as long as the following are selected:

- *Feature Modeling*
- *Colligens*
- *FeatureIDE*
- *FeatureIDE extension for AHEAD*

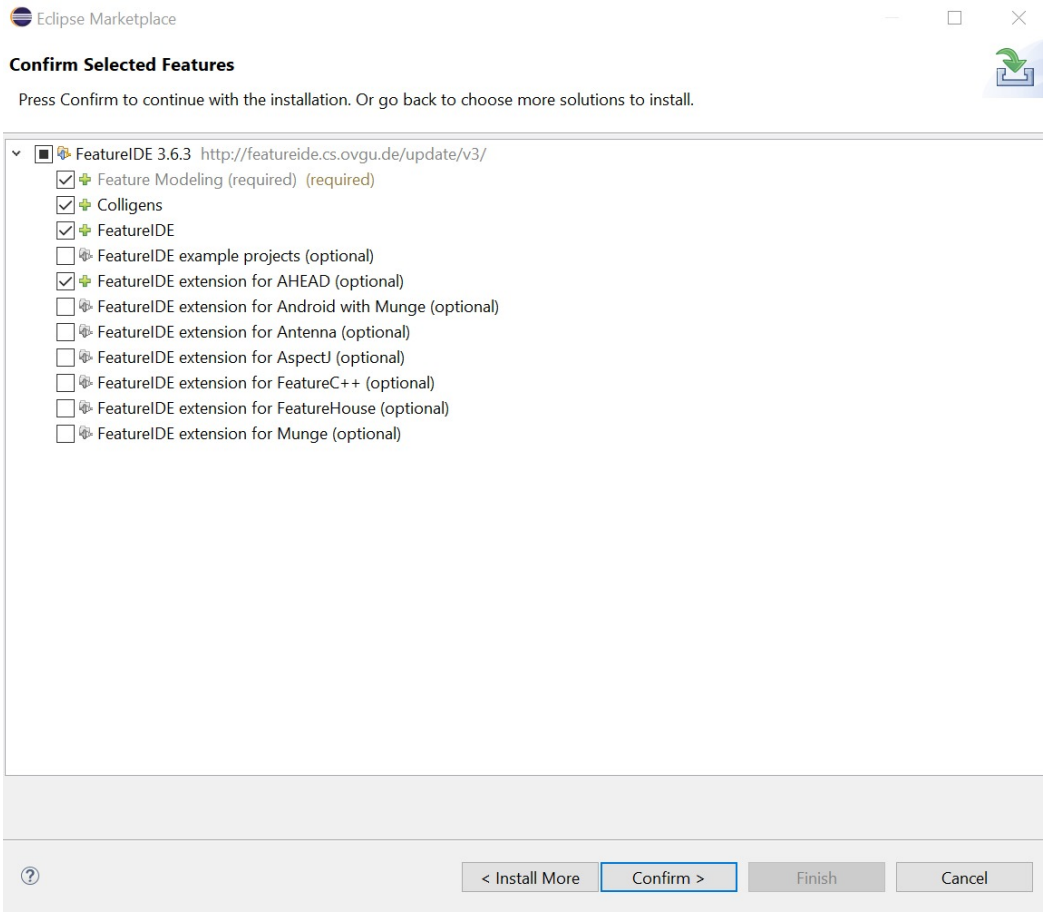

Figure 3: Install FeatureIDE plug-in *Confirm Selected Features* pop-up

2.4 Eclipse PDE Plug-in for Eclipse IDE

This plug-in is required. It allows the user to interact with the extension code as a developer which is required in order to download the project and run it.

Go to the *Help* tab on the top of the Eclipse software, select *Eclipse Marketplace*, search *Eclipse PDE* and press install. Then follow all the steps and accept terms as requested.

Note: To make sure it is the right plug-in, currently the logo is the Eclipse logo, the full title of the plug-in is *Eclipse PDE (Plug-in Development Environment) latest*, and the authors are Eclipse and EPL.

2.5 Git

The plug-in extension code is stored in a git repository. Hence, the user is required to have git previously installed. If this is not the case, please follow the instructions under the user's operating system at https://git-scm.com/book/en/v2/Getting-Started-Installing-Git[1].

3 Downloading FeatureIDE Plug-in Extension Code

3.1 Clone Git Repository

The first step required by the user is to clone the git repository which contains the plug-in code. To do so, open the computer file system, go to the directory where you have the eclipse workspaces. If you do not yet have an eclipse workspace folder, create one in the user directory. Now, open the terminal/command line in that location, type the following command, and press enter:
git clone https://github.com/marianaBonito/FeatureIDE_Extended_Urls.git

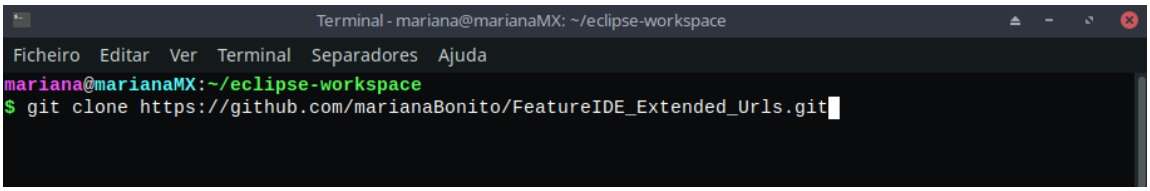

Figure 4: Illustration of the command in a Linux terminal

This requires internet access and may take some time. Once it is finished the user may close the terminal/command line.

3.2 Open Eclipse

Open the Eclipse application. It will ask which workspace to use. Press *Browse...* and select the file containing the plug-in code the user has just cloned. Unless the user has changed the name, the file will be called *FeatureIDE_Extended_Urls*.

Once the Eclipse application has been opened it should display a welcome tab as shown in figure 5.

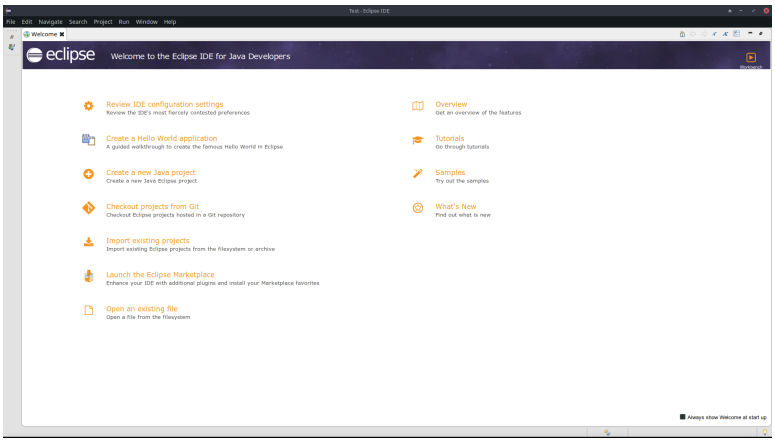

Figure 5: Eclipse welcome page

Note: Please do not be alarmed if the colors of the eclipse software are different as these are customizable in the eclipse preferences. They do not impact the outcome.

3.3 Import code

Now, press *File* on the top left corner of the Eclipse application and select *Import*. A pop-up should appear as shown in figure 6. Expand the *Plug-in Development* section, select *Plug-ins and Fragments*, and press *Next*.

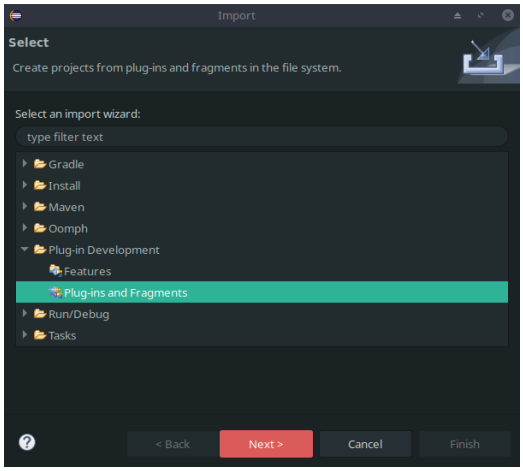

Figure 6: Plug-in development project

Now, the pop-up should look similar to the one in figure 7. The user should fill in the pop-up as follows:

1. On the *Import From* section, select the option *Directory*. Press the *Browse...* button and select the workspace folder containing the plug-in code (Same folder as the one used when opening Eclipse, probably called *FeatureIDE_Extended_Urls*).

2. On the *Plug-ins and Fragments to Import* section, choose *Select from all plug-ins and fragments found at specified location.*

3. Under the *Import As* section, select the *Projects with source folders.*

4. Finally, press the *Next* button.

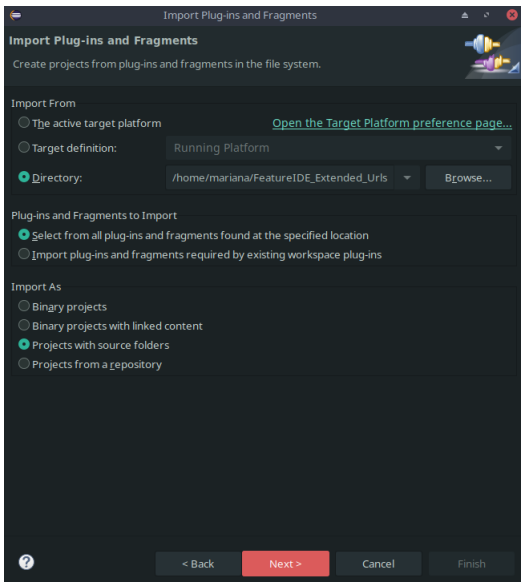

Figure 7: Import plug-ins and fragments pop-up

After, a pop-up similar to the one displayed in figure 8 should be visible. Make sure only the *Include fragments when computing required plug-ins* and the *Show latest version of plug-ins only* checkboxes are selected at the bottom. On the input text area to *Filter Available Plug-ins and Fragments* type *featureide* and press enter. Then press the *Add All* button, and finally press *Finish*.

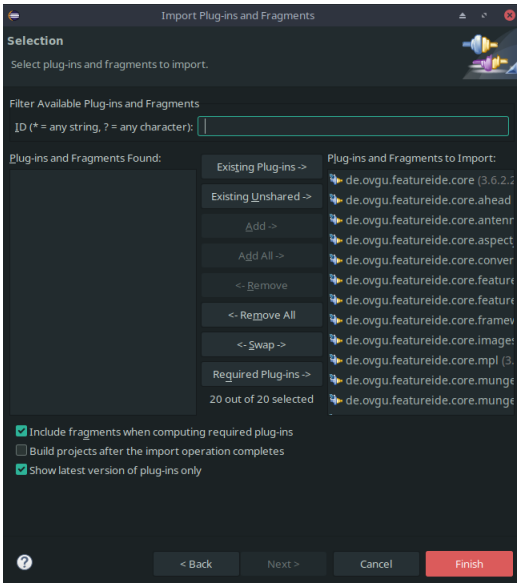

Figure 8: Plug-ins and fragments selection

Once the pop-up is closed and the progress bar on the bottom right corner of Eclipse has reached 100%, the eclipse window should look similar to figure 9.

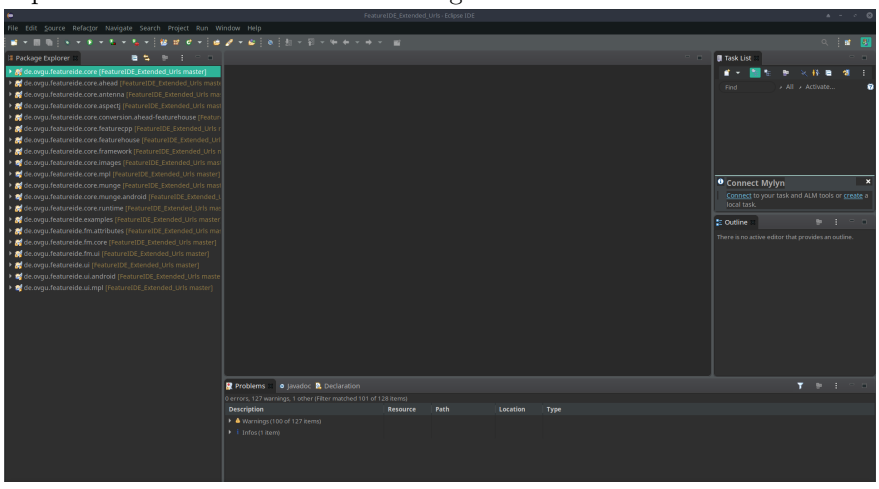

Figure 9: Plug-ins and fragments imported

Please make sure under the *Problems* tab at the bottom, there are no *Errors*, only *Warnings*, and *Infos*. If there are errors please delete all project files on the Package Explorer tab and re-do the steps from section 3.3. To delete all, press on one of the files, click *Ctrl* + *A*, Right-click and press *Delete*. When the pop-up is shown **do not** press *Delete project contents on disk*. Only press the *OK* button.

4 FeatureIDE Plug-in Extension Run and Use

This manual only covers the basic aspects of the Feature IDE plug-in to run and interact with the extension component. For a more in-depth understanding of how to use the rest of the FeatureIDE plug-in please consult the manual in https://github.com/FeatureIDE/FeatureIDE/wiki/Tutorial[4].

4.1 Running the Plug-in

In order to use the plug-in, first, the user must launch a new eclipse application that implements the plug-in code previously installed. To do so, right-click on *de.ovgu.featureide.core* package. This will most likely be the package at the top of the *Package Explorer*. Then, hover over *Run As* and select *1 Eclipse Application*. Figure 10 illustrates what this process should look like.

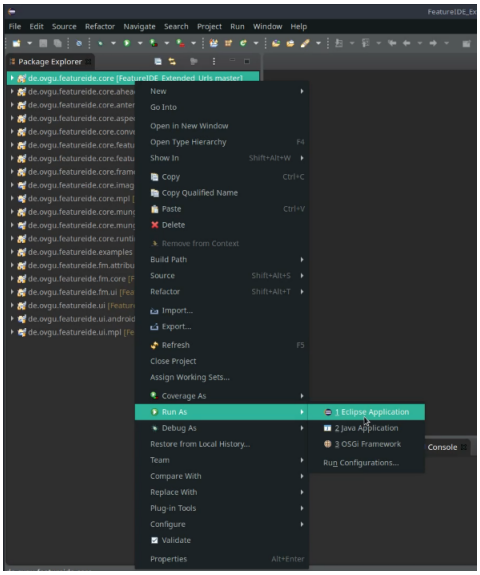

Figure 10: Run As Eclipse Application

Once you choose to run the code as an eclipse application, there may be a pop-up similar to figure 11 stating there is a validation problem. This has no impact on the plug-in execution and can be ignored. If it does pop-up, simply press *Continue*.

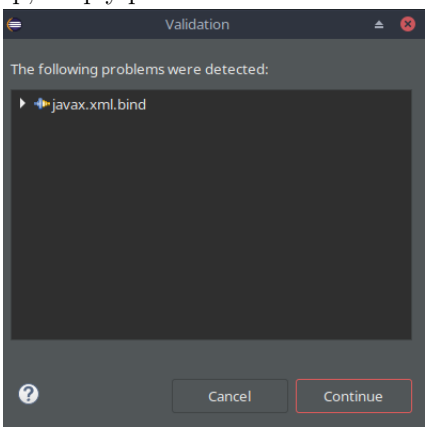

Figure 11: Irrelevant error pop-up

4.2 Creating New FeatureIDE Project

Now, there should be another instance of eclipse open with no projects on the *Project Explorer* tab. Both instances of Eclipse must be open at all times.

Note: If at any point the Eclipse application running the plugin code freezes, please go to the original Eclipse instance where the user right-clicked on the code to run it as an eclipse application, access the *Console* tab on the bottom and press the little red square button (shown in figure 12) to terminate the other eclipse instance.

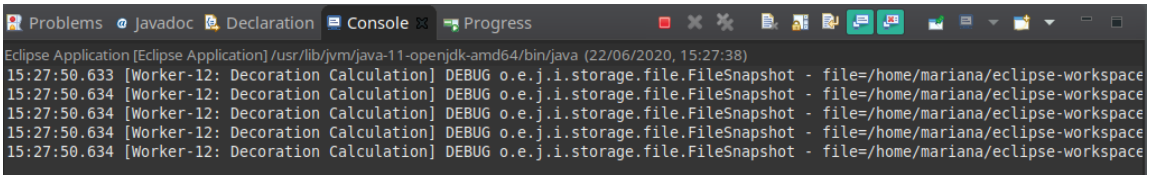

Figure 12: Terminate Eclipse application instance

To start using the plug-in, first, the user must create a new feature project. To do so, go to *File* on the top left corner of Eclipse, hover over *New*, and select *Project...*. A pop-up similar to figure 13 should appear.

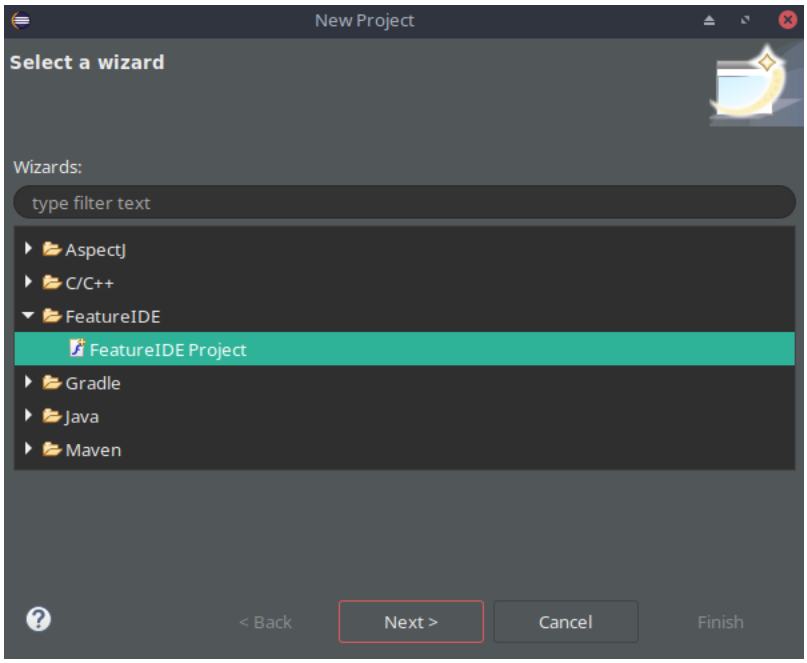

Figure 13: New FeatureIDE project

Expand the *FeatureIDE* option, select *FeatureIDE Project*, and press *Next*. Then, the pop-up should look something like figure 14. For this explanation, do not change anything and just press *Next*. This pop-up can be altered but requires a further understanding of the FeatureIDE plug-in not covered by this manual.

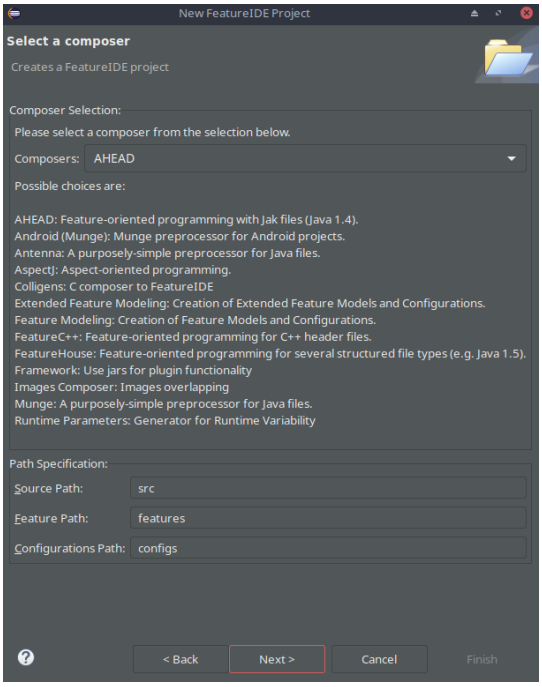

Figure 14: New FeatureIDE project select composer

The pop-up will now look like figure 15. The only thing required by the user is to input the desired project name in the input text area labeled *Project name* and press the *Finish* button.

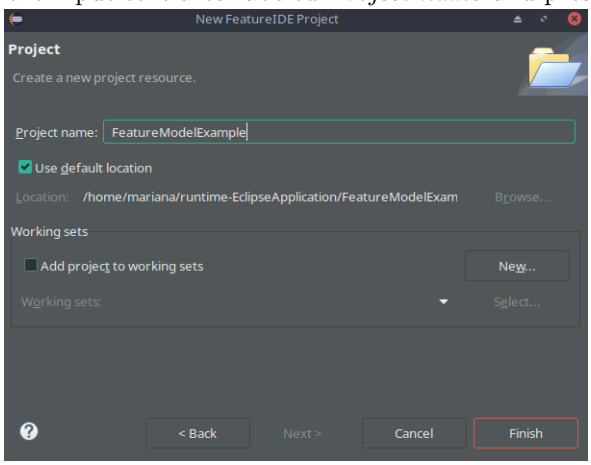

Figure 15: Assign FeatureIDE project name

A pop-up similar to the one displayed in figure 16 may appear now or later. It is not a requirement but it is suggested, that once it appears, the user should press *Yes*.

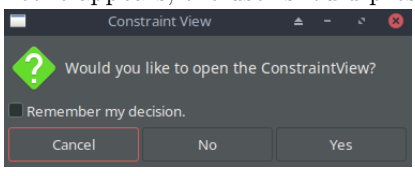

Figure 16: Constraint view pop-up

4.3 Using the Plug-in Extension

As mentioned, this manual does not explain how to use the standard components of the FeatureIDE plug-in. Instead, there is a link available at the beginning of section 4 to an instructions manual of the FeatureIDE plug-in.

The *Project Explorer* tab should contain the FeatureIDE project created by the user. It should look similar to figure 17.

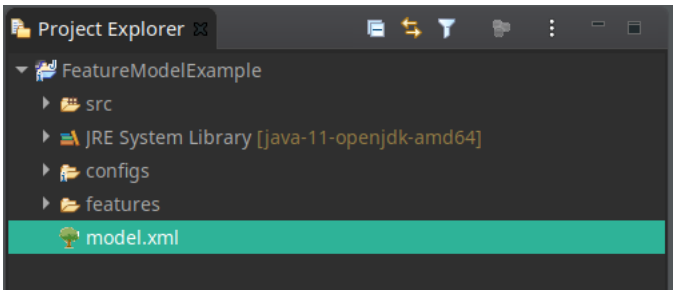

Figure 17: New FeatureIDE project created

If the project is not expanded, do so. Double click on the model.xml file. The feature model editor should display the base features as presented in figure 18.

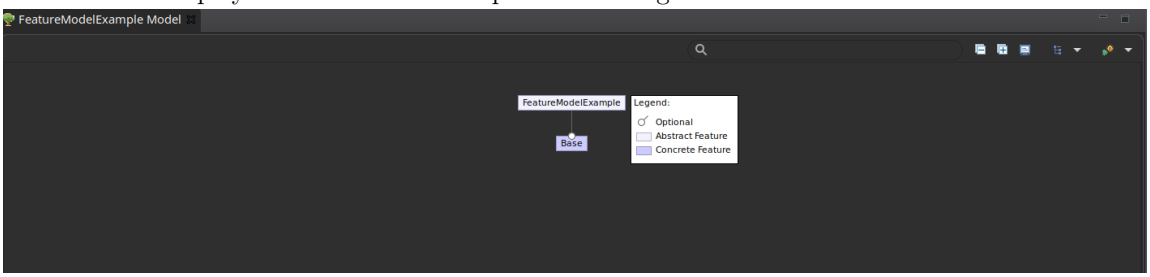

Figure 18: New feature model displayed

To start using the plug-in extension, right-click on a feature. Figure 19 exemplifies what the context menu should look like. Currently, there are no URLs associated with said feature.

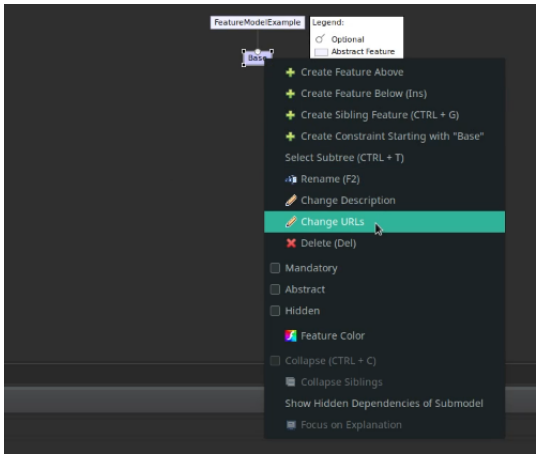

Figure 19: Right-click on a feature

4.3.1 Add/Change URLs of Each Feature

To associate one or more URLs to a specific feature, press the *Change URLs* option on the right-click menu.

Once the user selects said option, a pop-up as illustrated in figure 20 should appear.

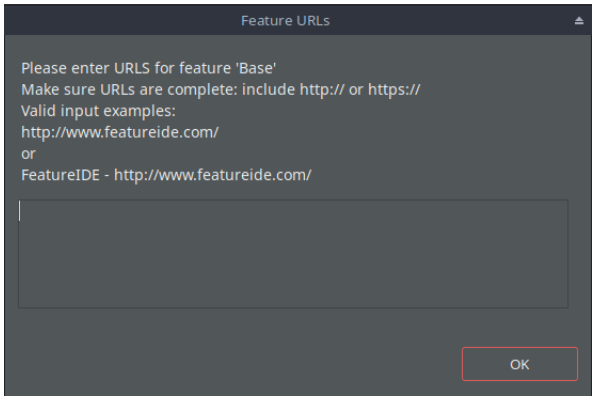

Figure 20: Change URLs pop-up

If the user hovers with the mouse over the pop-up description, there is a further explanation of the format the URLs must be inserted. The tool-tip is displayed in figure 21.

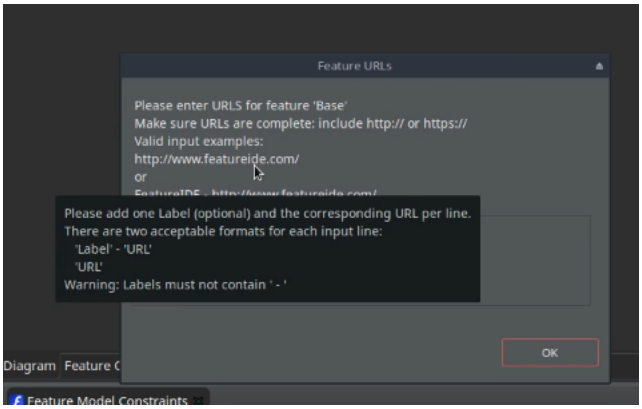

Figure 21: Change URLs pop-up explanatory tool-tip

If the user wishes to add a URL to the selected feature, the input rule in the dialog box are briefly explained in the pop-up description and tool-tip. However, a more in-depth description of the rules is described below:

1. Each line is for one URL. **Do not** split the URL into two lines, or add two URLs in the same line.
2. If there are blank lines between URLs these lines will be ignored
3. There are two acceptable formats to associate a URL to a feature. The user may wish for the URL to be listed with a label or not.
   (a) If the user wishes to associate a label to a specific URL:
      i. The label and the URL **must** be in the same line.
      ii. The label and the URL **must** be separated by " - ". This is space, dash, space. From now on, this pattern will be referred to as a *separation pattern*.

iii. There can only be one of these patterns (“ - ”) per line. Meaning, the label cannot contain a separation pattern in its name. Only to separate the URL from its label.

iv. Each line containing a URL and a label must be in the following structure: label, separation pattern, URL.

An example of a valid URL input with a label is shown in figure 22

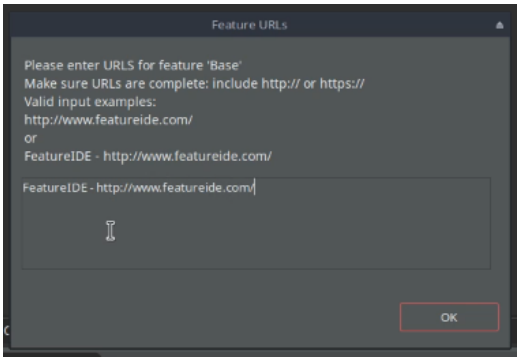

Figure 22: Change URLs pop-up correct input example with a label

(b) If the user does not wish a particular URL to have a label all the URL line should contain is the URL itself.

4. The user should make sure the URL input contains the **whole** URL. For example, typing www.google.com is not a valid URL and will not work properly. Instead, the user should write https://www.google.com/.

Once the desired URLs are written in the dialog box, press *OK*. The plug-in will attempt to validate the URLs. Possible errors that may occur are further explained in section 4.3.3.

4.3.2 Viewing Feature URLs

Once the features have URLs associated with them, the user may wish to view and open them. To do so, there are two options:

1. **Right-click on the feature** to open the context menu. Whenever there are one or more URLs associated with a feature the menu will be similar to the one presented in figure 19, but with a small alteration. There will now be an entry labeled *List URLs*. If the user hovers or presses on it, it will list the URLs associated with the feature. The list will display either the label of the URL (if it was given one) or the whole URL. Figure 23 demonstrates how the URL input in figure 22 would be displayed on the right-click menu.

2. There is also a shortcut in place that allows the user to list the URLs associated with a feature without having to access it through the right-click menu. Simply **click the desired feature with the mouse and press *Ctrl + U***. This, like the *List URLs* in the right-click menu, will display either the label of the URLs or the whole URLs. Figure 24 demonstrates how the URL input in figure 22 would be displayed on the shortcut list.

On either option, if the user wishes to open the URL on their default browser, all that is required, is to click on the desired URL/ label and the plug-in will open the window with said URL.

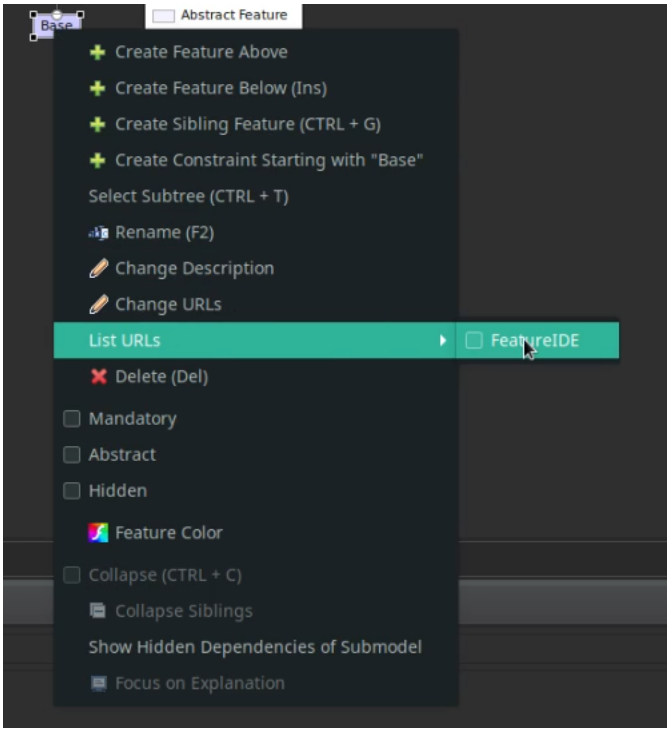

Figure 23: Right-click list feature URLs example

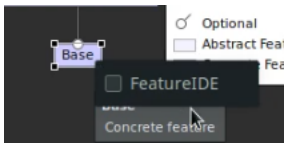

Figure 24: *Ctrl* + *U* list feature URLs example

4.3.3 URL Validation Errors

When inputting the desired URLs for a specific feature, there are certain errors the plug-in is prepared to deal with. The URL validation is executed once the user presses the *OK* button. It relies on an internet connection, but as the user might wish to add URLs while offline that should not be a limitation. Obviously, some functionalities such as opening the URL on a browser will not be available. Nevertheless, the plug-in must still be operational without internet access. Hence, the possible errors and how they are handled by the software are listed below:

1. Errors when inputting URLs in the *Change URLs* pop-up:

   (a) If the user's computer has constant access to the internet, all rules mentioned in section 4.3.1 are verified. If any of the rules are not met, the feature's *Change URLs* pop-up will display the number of URLs with errors, stating which URL is the first one with errors. It will also not allow the user to close or press *OK* while the URLs are not corrected. It will show the error information until there are no errors left. Figure 25 illustrates an example with 4 URL lines.

   The first line is a correct example, The second has more than one separation pattern, the third has no URL and the fourth has a label and an incomplete URL. As shown,

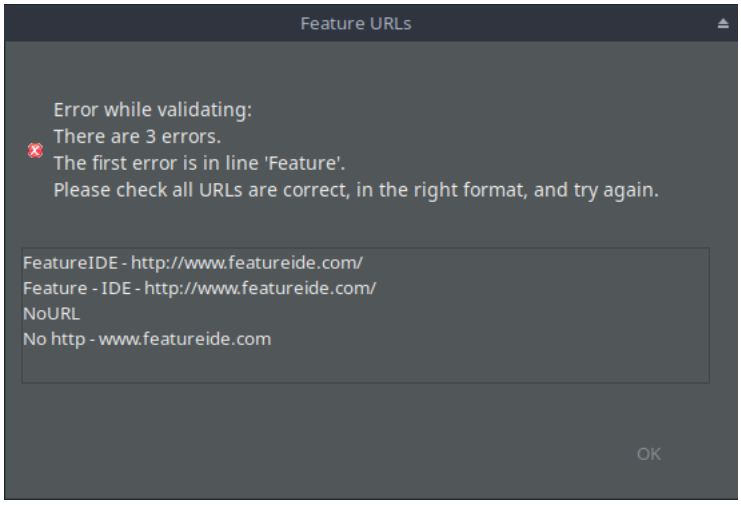

Figure 25: Change URLs pop-up incorrect input example

the error message states there are three lines wrong and the first one is the one with the label "Feature" (second line).

(b) If the user's computer had internet at the beginning of the validation but lost it in the meantime, a pop-up like the one displayed in figure 26 will appear.

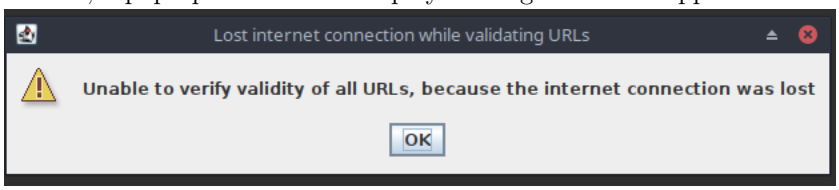

Figure 26: Warning pop-up: Lost internet connection while validating

Once the user presses *OK* two things may happen. If there where errors detected before the internet connection was lost, the user must correct said errors and press *OK* again. If there where no errors detected before, the pop-up will close and assume the content is correct.

(c) If there was no internet connection since the user pressed *OK* on the feature's *Change URLs* pop-up, the software will not validate any of the content. Instead, it will display a pop-up like the one in figure 27, and assume all of the content is correct. It leaves the handling of the possible errors for later when there might be an internet connection.

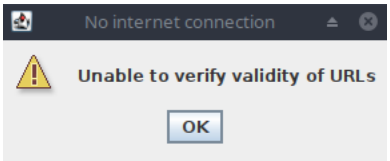

Figure 27: Warning pop-up: No internet connection to validate

2. Errors when viewing the URL lists:

(a) Assume there was no internet to validate the input URLs in the feature's *Change URLs* pop-up. When the user wishes to view all the URLs associated with a specific feature, the

list will present all URLs without knowing if it is a valid URL or not except for the URL lines that contain more than one separation pattern. For example, figure 29 illustrates what would be listed if in the feature's *Change URLs* pop-up, had been written what is displayed in figure 28 and there was no internet connection to verify the information was correct.

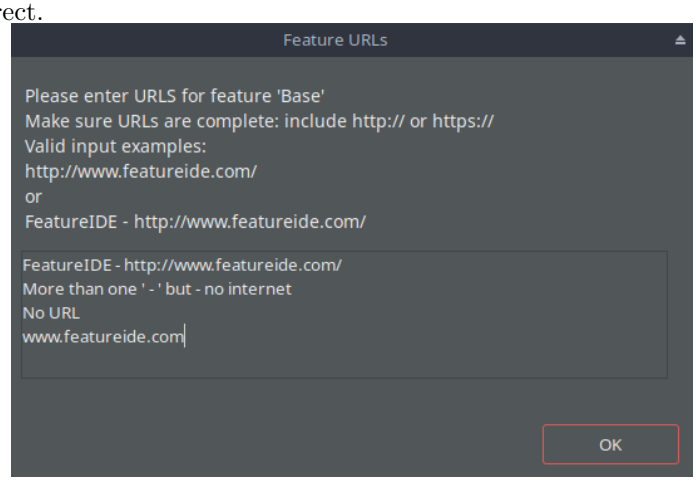

Figure 28: Change URLs pop-up incorrect input example

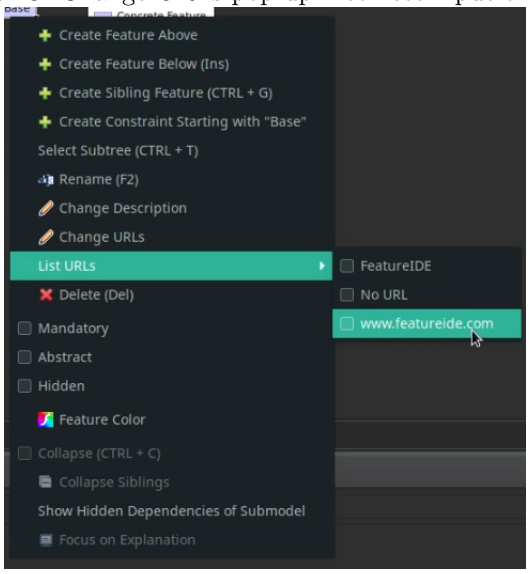

Figure 29: Right-click list feature URLs with no internet when validated

(b) When clicking on a URL from the list, if the user has no internet connection, a pop-up similar to figure 30 will appear. If the user tries again and there is still no internet, the pop-up will keep appearing. However, once there is internet again, if the user clicks on a URL, it will open the browser with the chosen URL.

(c) When clicking on a URL from the list, if the associated URLs were not validated, the user has access to the internet, and the chosen URL is not valid, A pop-up similar to figure 31 will appear. If the user tries again without correcting the URL, the pop-up will keep appearing. However, once it is corrected if the user clicks on the URL, the software will open it in the browser.

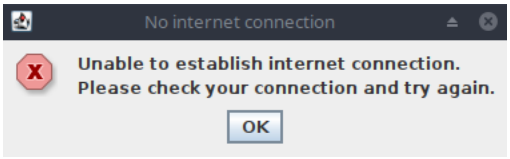

Figure 30: Error pop-up: No internet connection to open URL

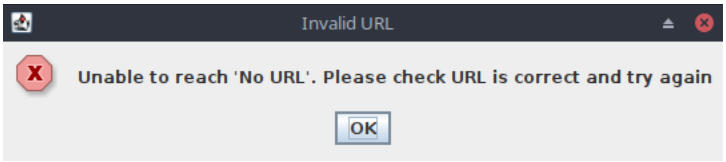

Figure 31: Error pop-up: URL was not validated but is unreachable (probably wrong)

5 Download Already Existing Projects

As this manual is part of a student investigation, this section describes how to access and view the two domain models, part of the author's project, containing URLs for some features. In this case, the URLs are links to multimedia which are used to exemplify the objective of a particular feature.

Firstly, access the computer files. Once the user has run the plugin code, as an eclipse application, there should be a folder on the same location as the *FeatureIDE_Extended_Urls* folder called *runtime-EclipseApplication*. Go into that folder and open the terminal/command line in that location. Now clone both of the feature models' git repositories by typing:
git clone https://github.com/marianaBonito/HistoricalSoundscapesFeatureModel-BO.git, pressing enter, then
git clone https://github.com/marianaBonito/HistoricalSoundscapesFeatureModel-FO.git, and finally pressing enter again.

This requires internet access and may take some time. Once it is finished the user may close the terminal/command line.

Now, go to the eclipse application running the plug-in extension code. Create a new project as explained in section 4.2 but when the user is requested a project name, insert *HistoricalSoundscapesFeatureModel-BO*. Now, expand the project on the project explorer, and open the model.xml file. A feature model similar to figure 32 should appear.

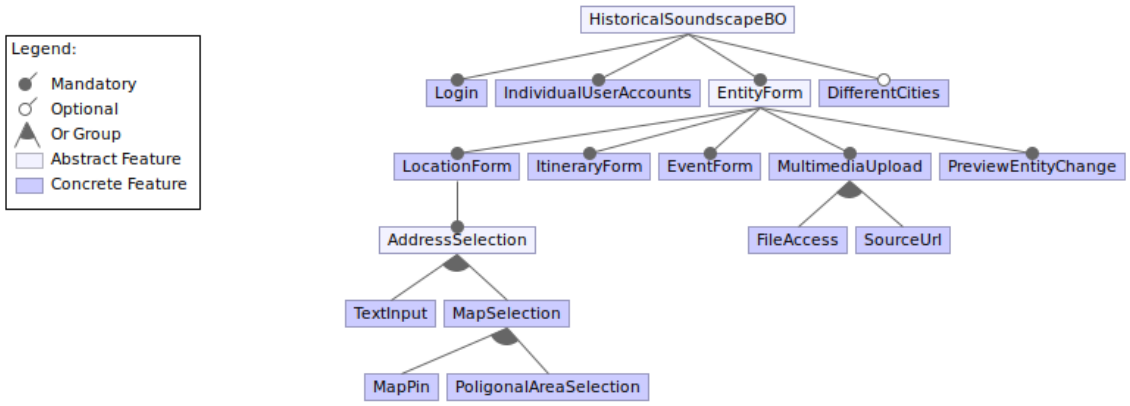

Figure 32: Author's project, back-office feature model

Now repeat the process of creating a project but this time, name it *HistoricalSoundscapesFeatureModel-FO*. Again, expand the project on the project explorer, open the model.xml file. A feature model similar to figure 33 should appear.

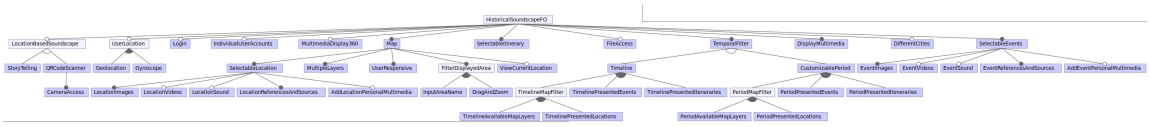

Figure 33: Author's project, front-office feature model

Once both models have been downloaded, they are ready to use and view as explained in section 4.3.

References

[1] Chacon, S., & Long, J. (2019, December 22). 1.5 Getting Started - Installing Git. Retrieved from https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

[2] Eclipse Foundation, Inc. (2020, June). Eclipse Installer 2020-06 R. Retrieved from https://www.eclipse.org/downloads/packages/installer

[3] FeatureIDE. (2020). Retrieved from https://featureide.github.io/

[4] Meinicke, J. (2017, June 08). FeatureIDE/FeatureIDE. Retrieved from https://github.com/FeatureIDE/FeatureIDE/wiki/Tutorial

[5] METOP GmbH. (2015). FeatureIDE. Retrieved from http://www.featureide.com/

[6] Oracle. (2020, April 10). Java SE 11 - Installation Guide. Retrieved from https://docs.oracle.com/en/java/javase/11/install/index.html