Sofa 2.0

Repository Editor Eclipse plug-in.

User's guide

Maksym Nesen

Chapter 1. Introduction. 3
Chapter 2. System requirements and installation 5
Chapter 3. Working with the plug-in. 6
3.1 Checking proper install of the plug-in. 6
3.2 SOFA 2.0 perspective 8
3.3 Direct work with repository 9
3.3.1 Frame centric and architecture centric modes of the repository view. 9
3.3.2 Create frame. 10
3.3.3 Create architecture 12
3.3.4 Create interface 12
3.3.5 Create subcomponent 16
3.3.6 Links 19
3.3.7 Frame properties 20
3.3.8 Architecture properties 21
3.4. Working with cushion extension 21
3.4.1. Creating new cushion project 21
3.4.2. Package explorer, cushion view, multi-page editor 23
3.4.3 Create new interface type 23
3.4.4 Create new frame 25
3.4.5 Create new interface 27
3.4.6 Create new architecture 28
3.4.7 Create new subcomponent 30
3.4.8 Committing to the repository 31
Chapter 4. Uninstall the plug-in 33
Chapter 5. Support, troubleshooting, further information 35
Chapter 6. Open issues 35

Chapter 1. Introduction.

This guide is aimed to user who has at least basic knowledge of the Sofa 2.0 system. The plug-in is developed as the extension for the Sofa 2 project which was developed by the students and researchers team. The plug-in supports work with repository and cushion parts of the Sofa 2 system. To get more knowledge about the Sofa 2 project reader should refer to the http://sofa.objectweb.org/

The plug-in is developed accordingly to the component model requirements. It supports creation and editing of all properties defined by the Sofa 2.0 meta-model. Also it provides 2 different kinds of creating entities in the repository. The first is direct work with repository part. The 2nd is generation of needed entities using the cushion extension of the Sofa 2.0 project. To know more about the cushion extension reader should refer to the http://sofa.objectweb.org/docs/cushion.html

Using the plug-in user can create frame and architecture entities almost in 1 click. Appropriate wizards will guide user through the process of creation needed entities. When user works directly with repository it is possible to create frame and place needed interface onto its borders. Interfaces can be of provided and required types. Then, when interfaces are placed, it is possible to edit interface properties and set needed name, version, connection-type, and also the most important – it is possible to write name, version, and signature of an interface-type which will be assigned to the interface. It is important to remember that interface-type should be set for each interface.

Then user can move to the creation of architectures. Each architecture must implement one or more frames. This is strictly defined by the meta-model. So, when creating an architecture, the appropriate wizard will ass to choose what frame should be implemented by the fresh architecture. When the frame is chosen it is possible to finish wizard and see the fresh architecture drawn on the draw area. When the architecture is created, all frames, which are implemented by it bypass their interfaces to be visible on the architecture's borders. Now user can create subcomponents inside the architecture. There is a wizard which helps to create needed subcomponent properly. If subcomponent is instantiated by frame which already has interfaces than all of those interfaces are visible on borders of the subcomponent. The same is valid if the subcomponent is instantiated by an architecture, which has interfaces, received from implemented frames.

When there are subcomponents with interfaces inside architecture, it is possible to draw links between interfaces. The current meta-model allows almost any linking between interfaces.

When working with the cushion extension of the Sofa 2.0 system, it is possible to generate needed architectures, frames, interface types using appropriate cushion commands. When a frame is generated, it is possible to edit it and add interfaces to its borders. And then all interfaces should have appropriate interface types defined. In this case all needed interface types must be generated before any interface is created. So, the entities generation sequence looks like this: firstly user should generate some interface types, then several frames and then several architectures. Then it is possible to add interfaces to frames and assign previously generated interface types to interfaces. And then it is possible to point which architecture implements which frame. Also it is possible to add subcomponents to the architecture. Then all changes should be saved and committed to the running working server. In fact, when an entity is generated using cushion, a trace of this entity is shown in the repository and is visible via the direct repository editor of the plug-in. But it is not recommended to change this trace. It is not completed locally, so when changes are done directly on the repository server, then

synchronization will be lost and it will be required to checkout changed entity to the local cushion project.

Chapter 2. System requirements and installation

The plug-in requires Eclipse IDE 3.2 or higher and Java 6.0 installed. Also it is required to install and run the Sofa 2.0 repository server. For more information please refer to http://sofa.objectweb.org/. Optional requirement is Apache Ant. No other plug-ins are required in order to install and run the sofaclipse plug-in.

The installation can be done in several ways: the current version of the plug-in can be downloaded as a local copy and installed as a local site from the Help -> Software Updates -> Find and install wizard. Also it can be installed using the same wizard but from the remote site (http://dsrg.mff.cuni.cz/~nesen/eclipse/update/). And finally it can be compiled and installed from source.
For more information about the installation procedure, please refer to the installation guide which should be supplied with this manual or to the online installation guide which is available from http://dsrg.mff.cuni.cz/~nesen/eclipse/.

Chapter 3. Working with the plug-in.

3.1 Checking proper install of the plug-in.

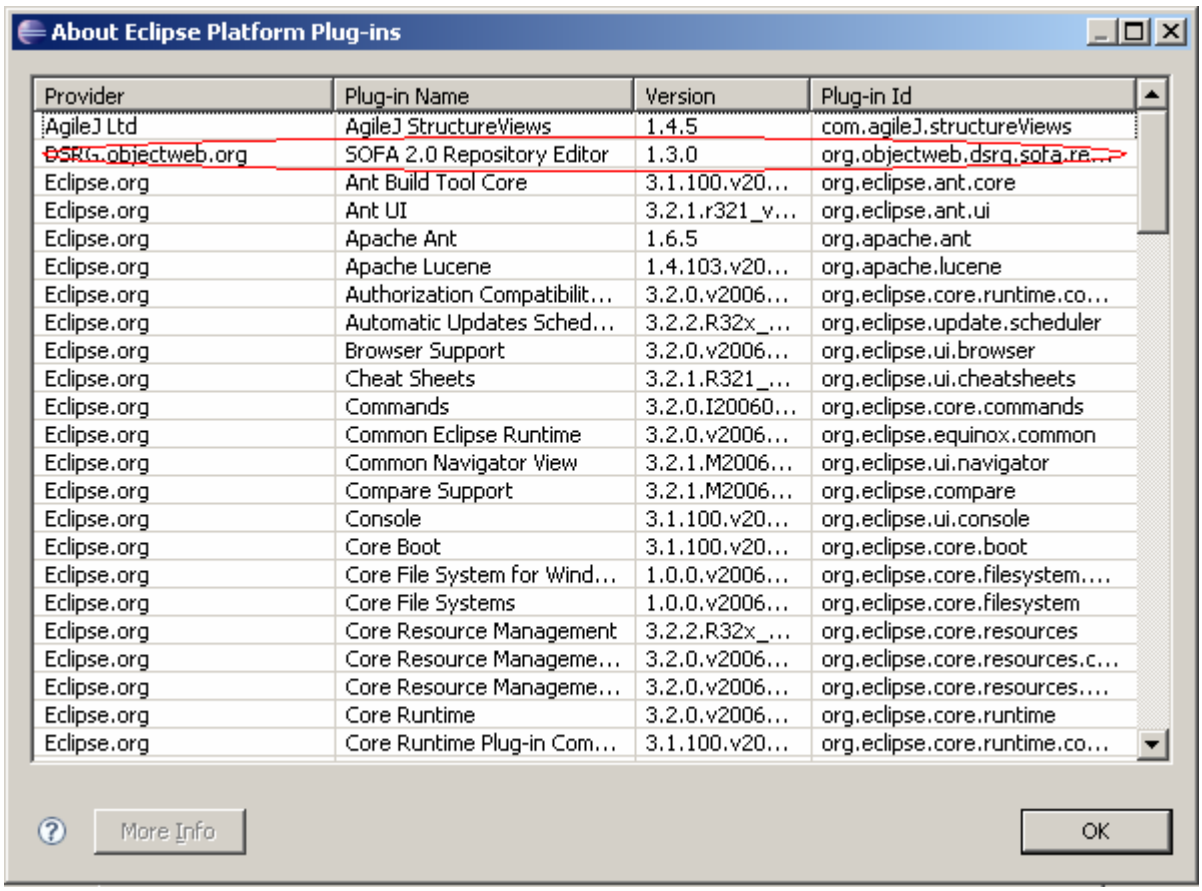
When the plug-in is installed it is required to check, if it is running properly. There are several contributions which are visible just on the start-up of the Eclipse IDE. When the plug-in is properly installed it contributes to the main toolbar. The new 'Save all changes to repository server' button is added to it. It should be visible between other buttons.

img 3.1 – toolbar contribution.

The next way to check the proper installation of the plug-in is to open Help -> About Eclipse platform and click the plug-in details button.

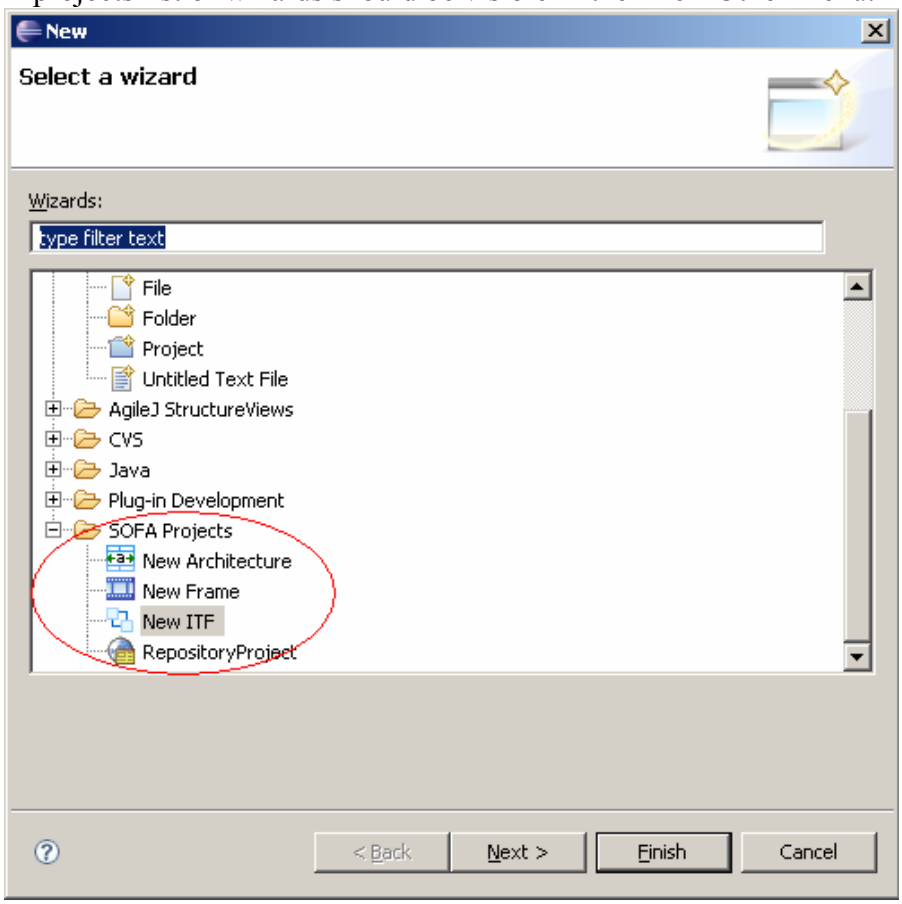
img 3.2 – About Eclipse dialog

Then in the list of plug-in it should be visible the SOFA 2.0 Repository editor loaded plug-in. The provider is dsrg.objectweb.org.



img 3.3 – About Eclipse Platform Plug-ins list.

Also the SOFA projects list of wizards should be visible in the File->Other menu.

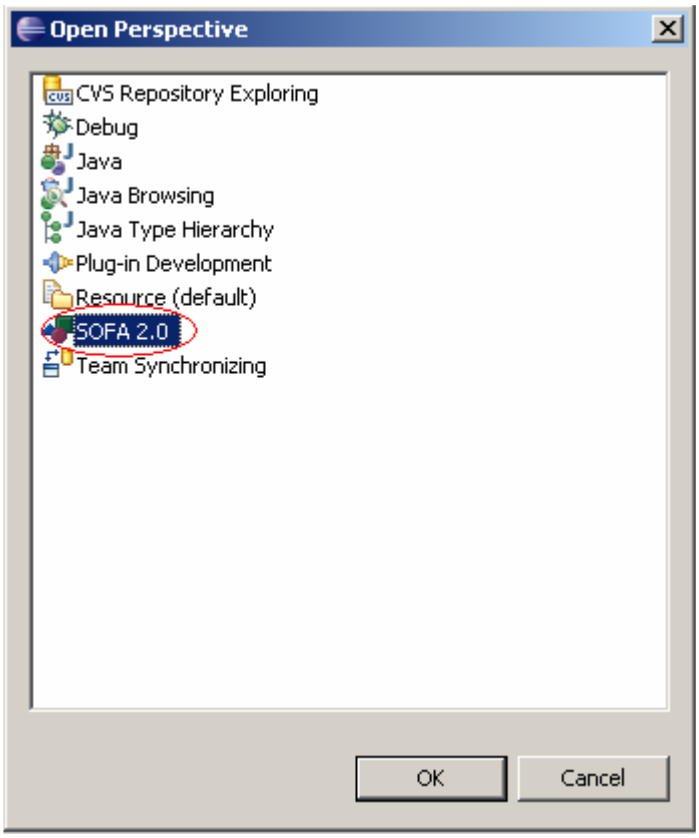


img 3.4 – SOFA Project list of wizards.

If all of those contributions are visible the plug-in is successfully installed and is ready to work. Now we can create the first repository project or the first cushion project. But it is required to open the SOFA 2.0 perspective in order to see all views properly located. The concept of Eclipse perspectives allows to group needed views using placeholders or direct location pointers. The only problem is that the editor area is one for all of perspectives. So if we use the editor area we should share it with all other perspectives.

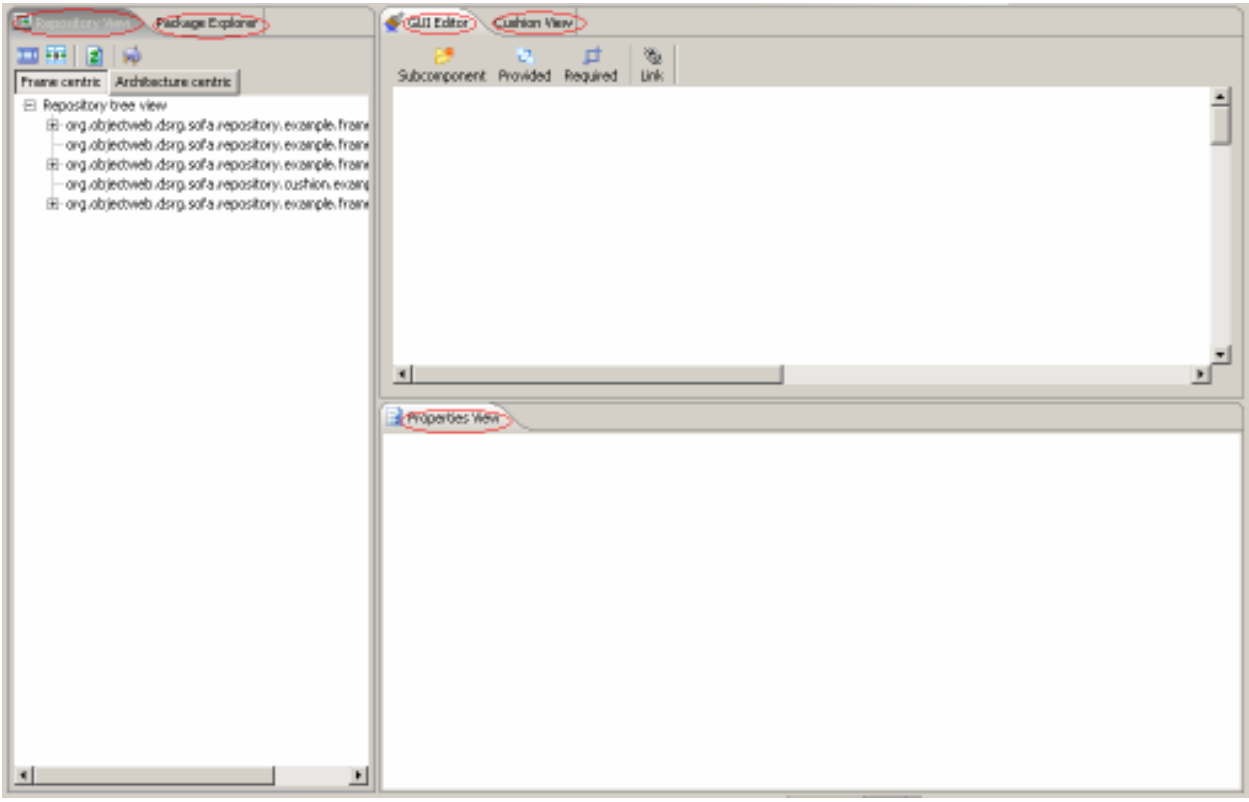
3.2 SOFA 2.0 perspective

In order to open any perspective in the Eclipse IDE it is required to use Window -> Open perspective dialog. It shows all available perspectives. We need to open the SOFA 2.0 perspective. Just follow those steps: Window -> Open perspective -> Other and in the opened dialog choose the 'SOFA 2.0' perspective.



img 3.5 – open SOFA 2.0 perspective

When the perspective is open, four views and package explorer are visible. In fact that is enough to work with the plug-in. When working with cushion extension it is also required to use the editor part, but it is not visible in the default perspective view to keep layout more accurate and to give more space to other views.
The opened views are: Repository view (tree view of architectures and frames stored on the repository server), GUI Editor – draw area, where all graphics representations of repository objects are drawn, Properties view – the view gives possibility to edit properties of entities which are selected at the draw area by mouse click or move. It is used only when needed to edit properties of object which are taken directly from the repository server, not from a cushion project. Cushion view – the area where all cushion generated entities are drawn. It has a bit different architecture than the GUI editor. To read more about this please refer to the developer's manual which should be supplied with this manual or refer to the online developer's documentation. The package explorer is open to browse and manage cushion projects.



img 3.6 - SOFA 2.0 perspective

3.3 Direct work with repository

In this chapter we will discuss the work with entities which are taken directly from the repository server. This includes creation of a frame, architecture, interfaces, and subcomponents, switching between architecture and frame centric views, and refreshing the repository tree. Also this includes linking of interfaces ob borders of subcomponents and architectures.

3.3.1 Frame centric and architecture centric modes of the repository view.

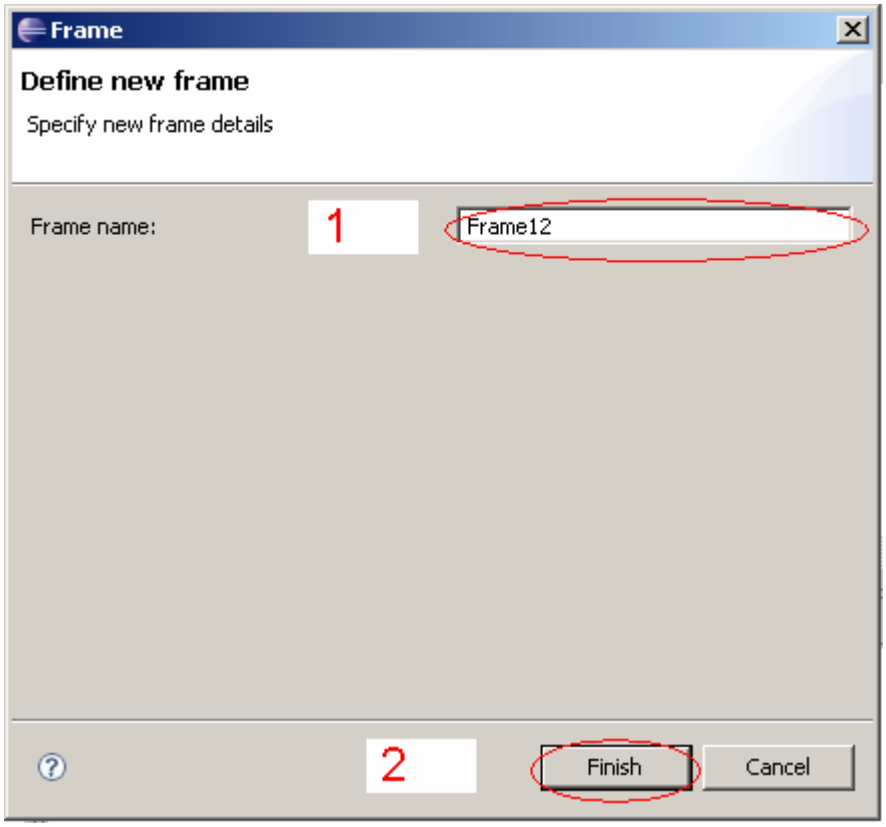
There are 2 modes of possible views of the repository tree – frame centric and architecture centric. This is done for possibility of multiple points of views on the repository tree. The main difference between those two modes is that the frame centric shows frames firs and attaches implemented architectures as nodes to parent frames. On the other hand the architecture view gives possibility to view all architectures first and see all implemented frames attached to architectures as nodes. For example, if there are several architectures generated via the cushion extension they do not implement any frame until they are finally committed from the local cushion files. Thus those architectures are not visible on the frame centric mode because the frame centric mode assumes that all architectures implemented at least one frame. But the architecture centric mode shows all architectures independently if they implement a frame or not. But in the architecture centric mode there are no frames which are not implemented by any of architectures.

img 3.7 – frame/architecture centric switch buttons

The frame centric mode is on by default.

3.3.2 Create frame.

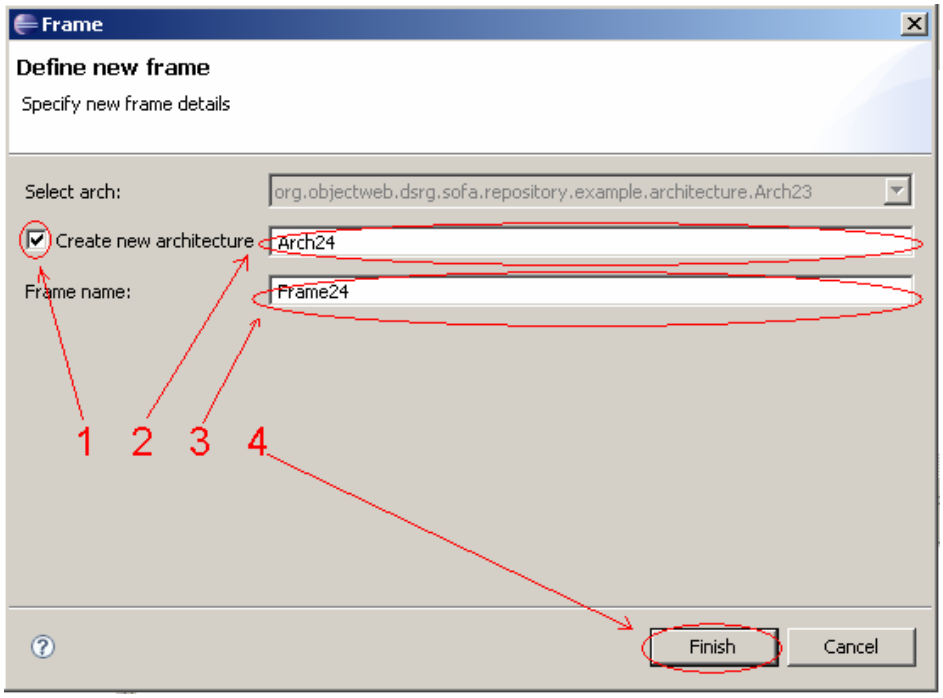
There are several ways to create a frame. This includes: create frame from the frame centric view – very simple way to create a frame, create frame from the architecture centric view and at the same time create new architecture which will implement new frame, and the last way is to create frame from the architecture centric view and select an existing architecture from the list of architectures, the selected architecture will implement new frame.
The first simple method of the frame creation should be done in this way: make sure the frame centric mode is the current repository view mode, then click the 'New frame' ( ) button which is located a bit higher then the 'Frame centric' (Frame centric) switch button and then follow the steps in the frame creation wizard.



img 3.8 – Frame creation wizard.

The Frame creation wizard includes two simple steps – the first is to type the name for the new frame and the 2nd is to click the 'Finish' button or press Enter key to finish the wizard. After frame is created, it is added to the list of frames at the Repository tree view and it can be accessed as well as all other frames from the tree.

The next way of the frame creation is to create a frame from the architecture centric mode and at once make it implemented by some architecture in order to be visible from the architecture centric view. To create frame it is required to follow these steps: make sure the architecture centric mode is on (Architecture centric), then click the 'New frame' button ( ) and follow the steps of the frame creation wizard.

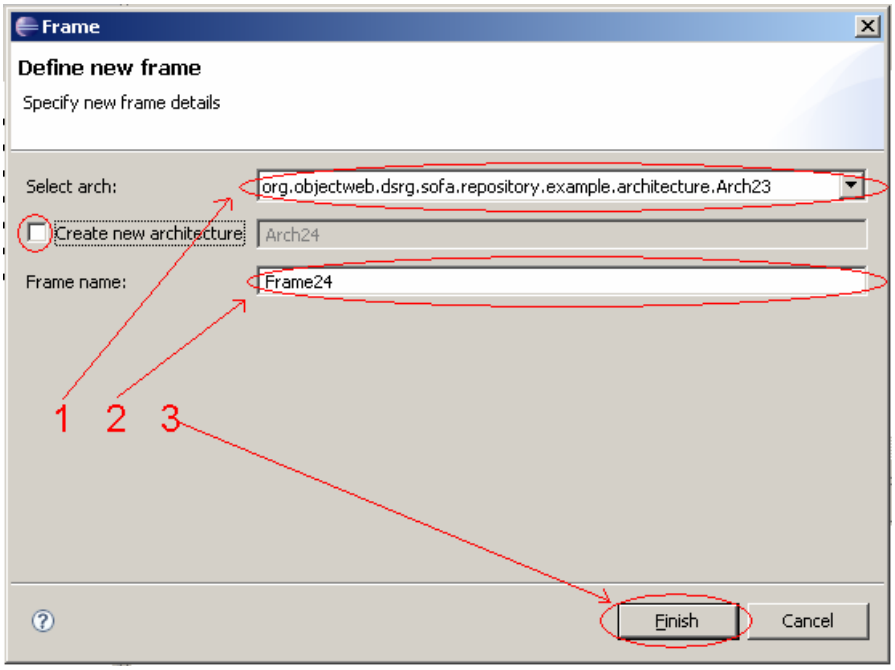


img 3.9 – Frame creation wizard (arch. centric mode)

This wizard includes 4 steps in order to create new frame along with the new architecture. First step is to tick the check box (the drop down list with names of architectures gets disabled), then type name of the new architecture, then type the name of the new frame. Then click finish or press the enter key to finish wizard and get the frame created. This frame will not be visible just from top because we are currently viewing the tree from the architecture centric mode. To see it we will need to expand the implementing architecture and then click on the name of the newly created frame. Then we will see its graphic representation on the draw area.

The last way of creating the new frame is shown below. Make sure the architecture centric mode is on (

), then click the 'New frame' button () and follow the steps of the frame creation wizard.



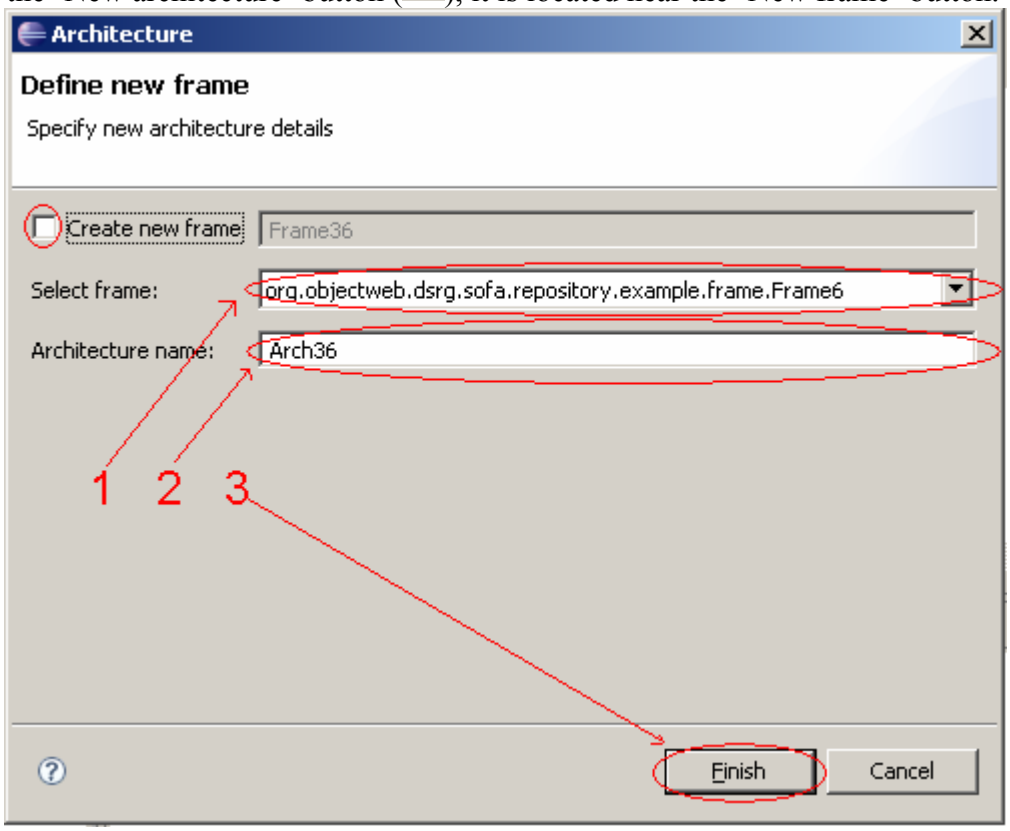
img 4.0 – Frame creation wizard (arch. centric mode)

This wizard can be done in 3 easy steps. Make sure the check box is unchecked and the drop down list with names of architectures is enabled (the text field to type architecture name is disabled). Then select the name of an architecture from the drop down list, then type in the name of the new frame and then click 'Finish' or press the enter key to finish the wizard.

This frame will be visible at the 2nd level of nesting in the repository tree view. It will be accessible after the appropriate architecture node is expanded.

3.3.3 Create architecture

Creating of architecture does not provide such many possibilities as for the frame creation. The wizard for the architecture creation is the same both for the frame and architecture centric modes. This is because according to the meta-model requirements any architecture which is created using direct repository access must implement a frame. That is why the wizard provides possibility to select or create a frame to implement for both views. To start creation procedure just click the 'New architecture' button ( ), it is located near the 'New frame' button.



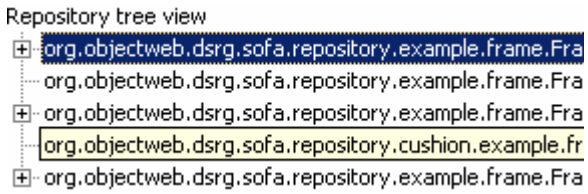
img 4.1 – Architecture creation wizard

The wizard gives 2 possibilities of the architecture creation – the first is to select frame to implement from the list, and the 2nd one is to create new frame to implement using the text field to type new name for the new frame. Any way each choice leads to the architecture creation.

3.3.4 Create interface

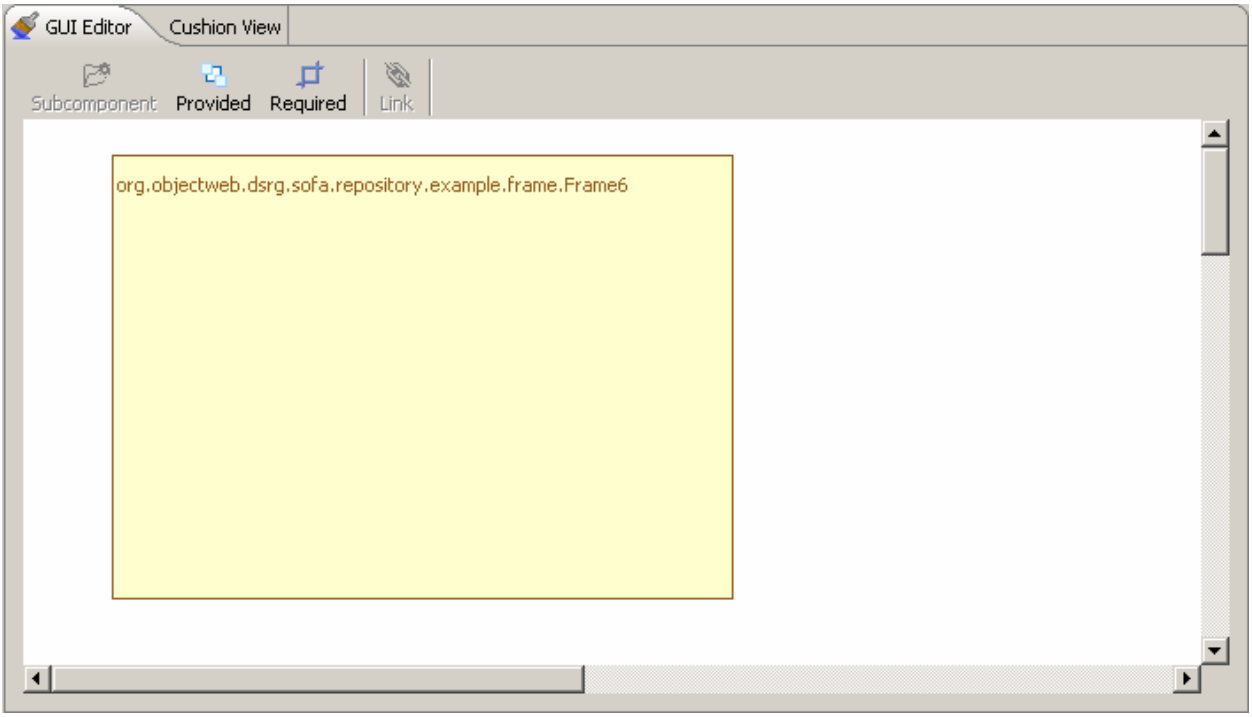
Interfaces can be created on frames, architectures and subcomponents. But the frame in fact is the interface container, and interfaces on architectures and subcomponents are just references to the frame which originally contains interfaces. This means that architecture which implements a frame with interfaces shows the same interfaces on its borders. The same is valid for a subcomponent. But on the other hand two different subcomponents can use interface from the

same frame to create links to different entities. One interface can link to the containing architecture and another can link to the next subcomponent.
Now let's start from the interface creation on the frame borders. Click on a desired frame do get it drawn on the draw area. Then on the toolbar over the draw area select and interface of the appropriate type. When the interface is selected, the cursor is changed to the cross. Point by this cross some where on the frame area. The interface will appear on the nearest border.



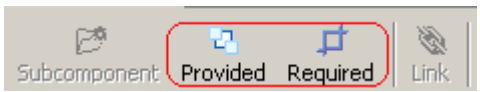
img 4.2 – selected frame in the list

The image 4.2 shows selected frame from the list of frames available in the repository. When any frame is selected, it is drawn in the GUI editor draw area.



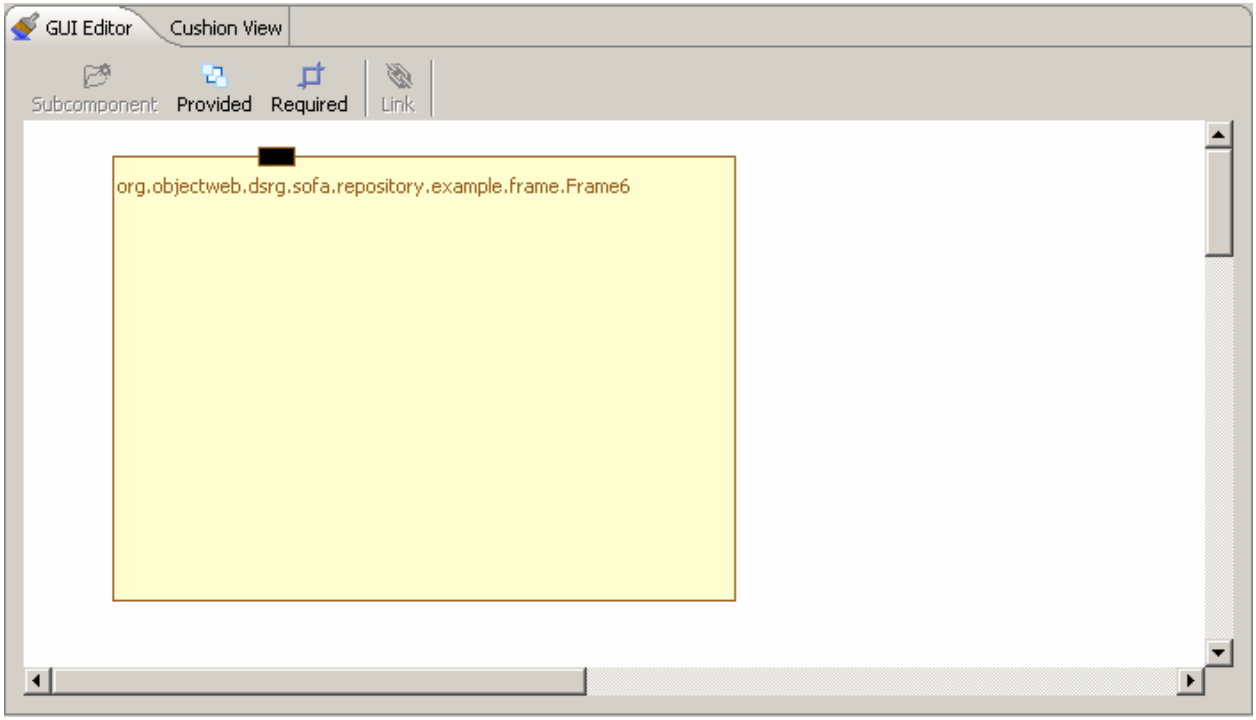
img 4.3 – drawn frame in the draw area.

Now, after frame is drown it is possible to put interfaces on it. The toolbar with interface buttons is visible on the screenshot.



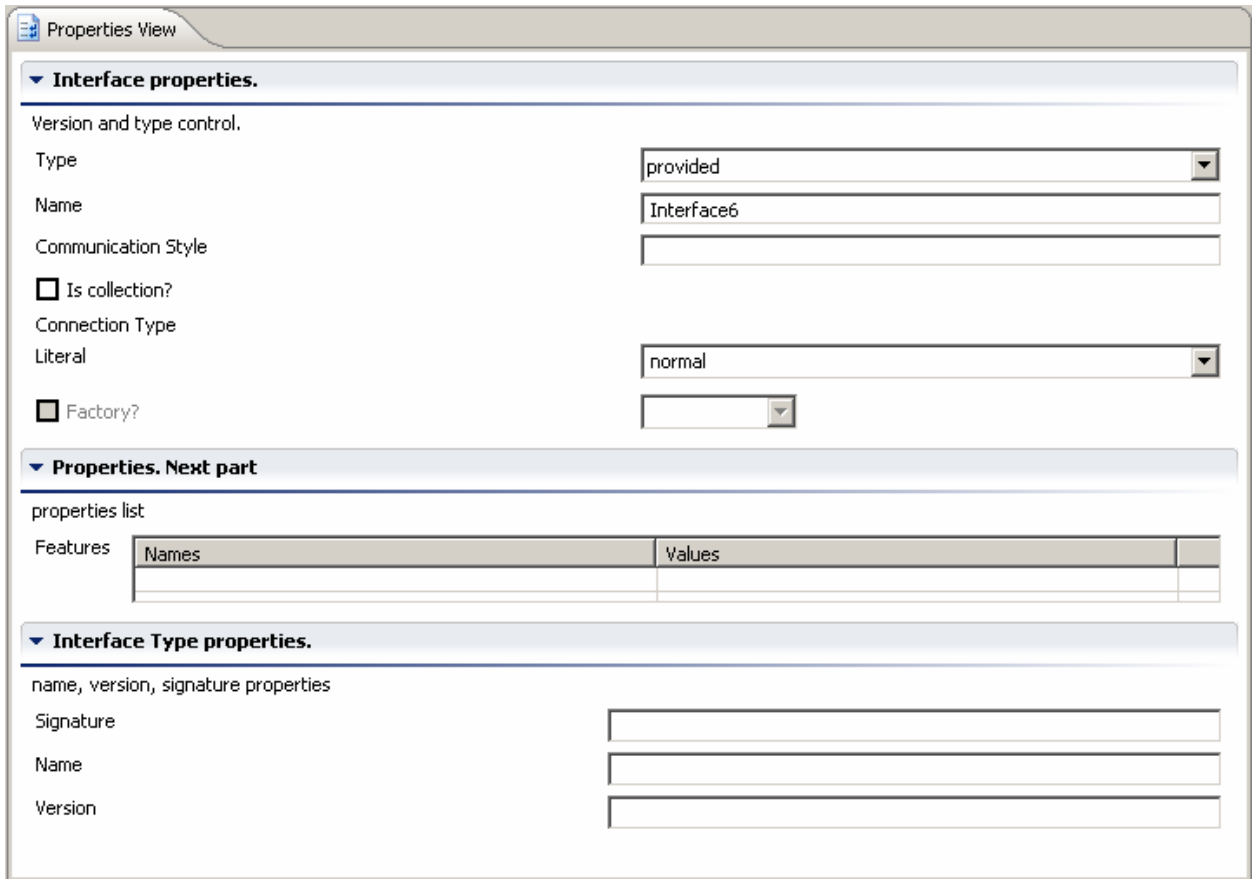
img 4.4 – the toolbar

The toolbar gives possibility to draw only interfaces. This is because frame does not support any other kinds of entities. Only interfaces can be put on frame. Please note, that the provided interface is drawn as a black rectangle and the required interface is drawn as the white one.



img 4.5 – the interface is drawn

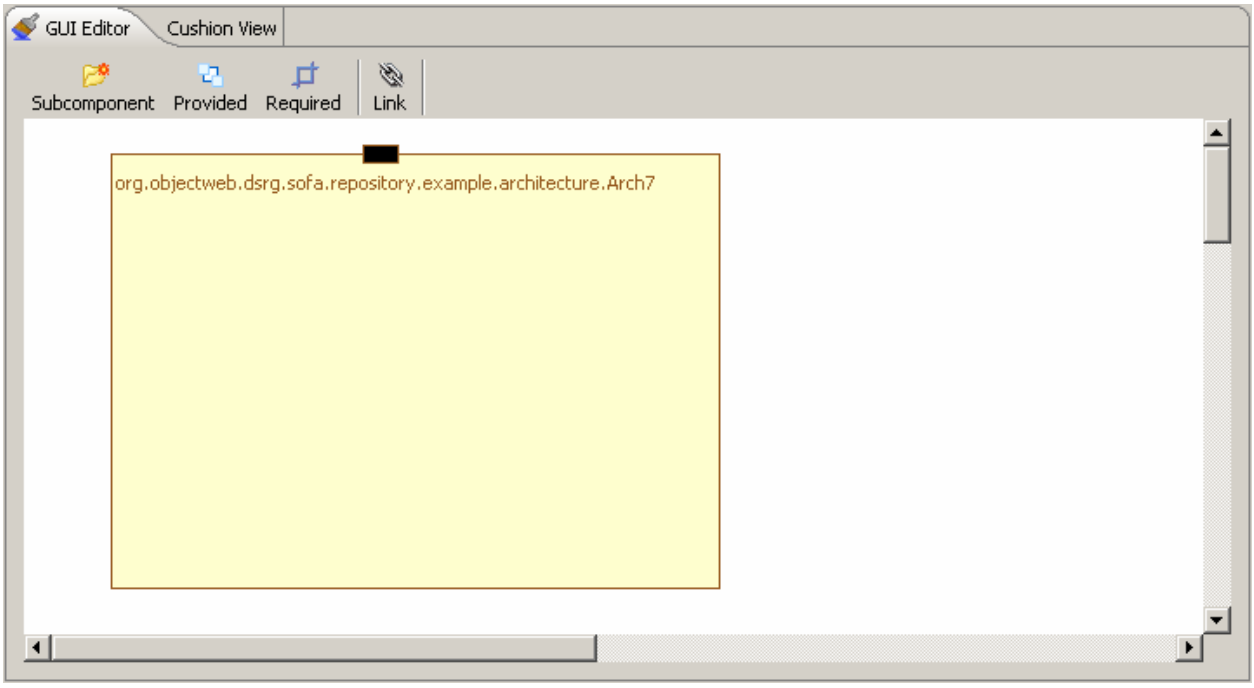
Now the interface is drawn. As soon as it is drawn we are able to change its properties. Interface properties includes name of the interface, interface's type (required or provided), communication style, collection property, factory property (which is available only when there are more than one interface on a border), list of features, and interface type properties – name, signature, and version of an interface type.



img 4.6 – list of interface properties (Properties view).

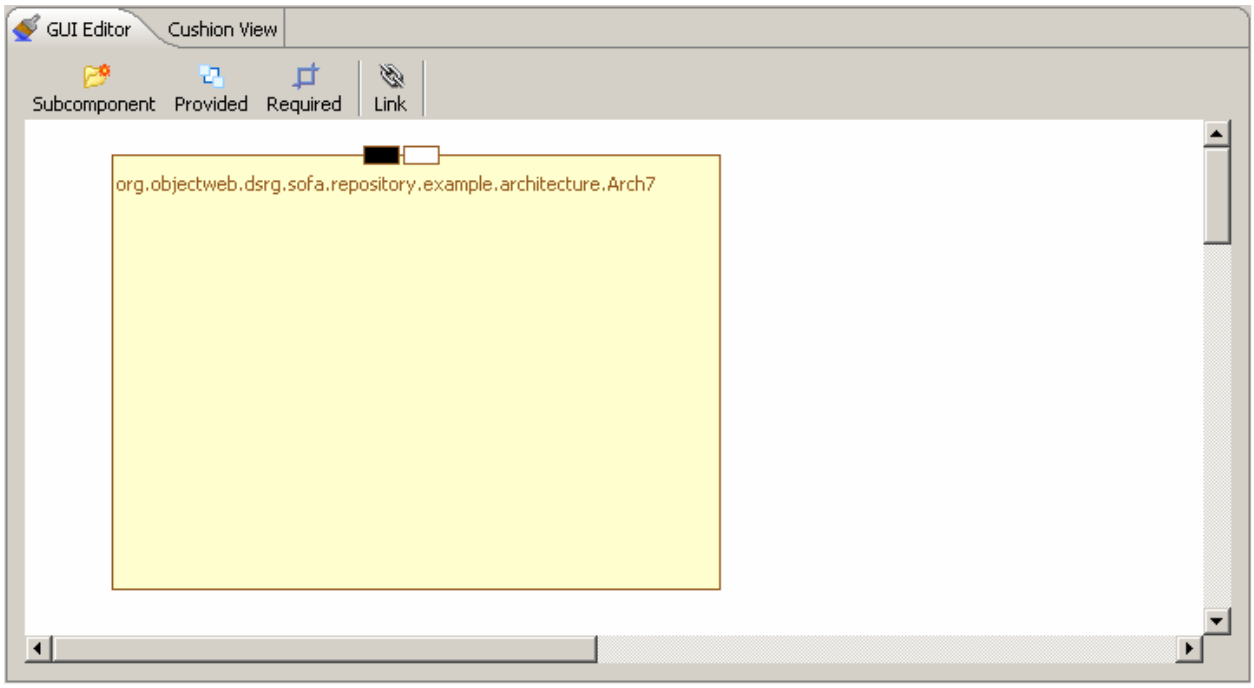
The main here are interface type properties. It is possible to save interface without any interface type, but then it will be useless for any further actions. That is why it is very preferable to set interface type just after the interface is created. Also the list of properties can be collapsed and there is no need to make it such a big. The screen shot is done just for demonstration purposes.

The next way to create an interface is to create it on architecture's border. Now it is required to select architecture from the list of architectures. Then perform about the same steps as for the interface creation on the frame's border.



img 4.7 – selected and drawn architecture.

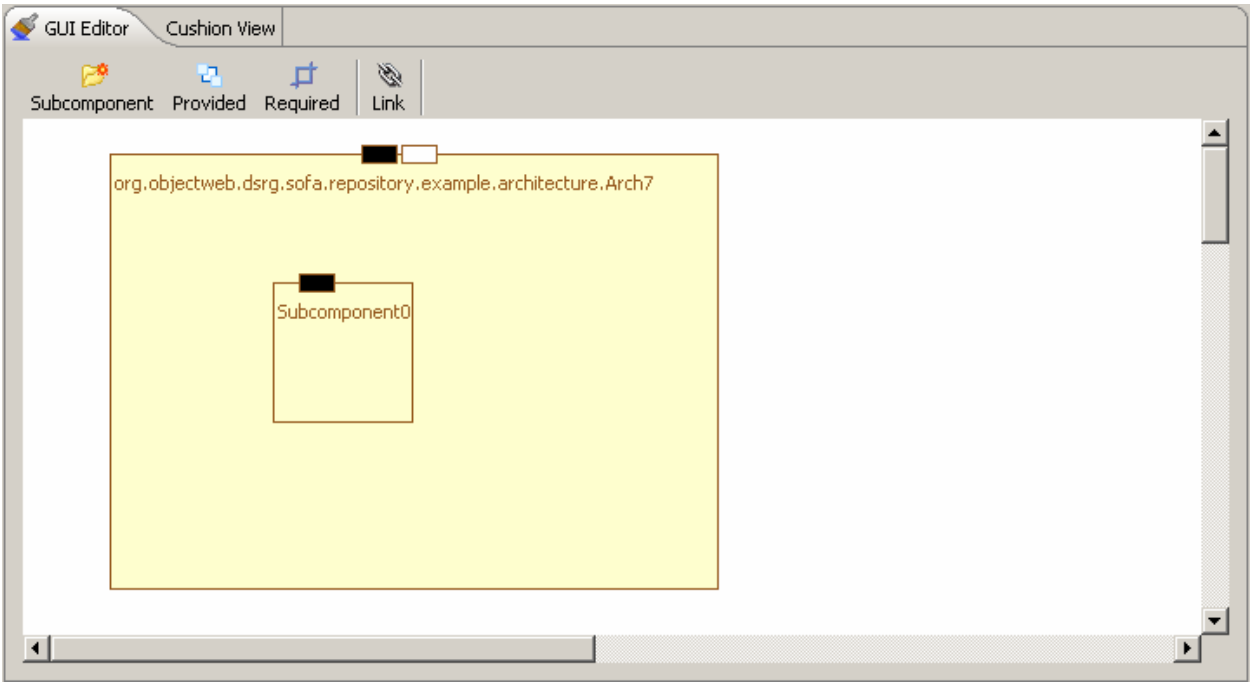
The architecture 'Arch7' is selected, it implements the frame 'Frame6', that is why already there is the interface on it. As you see, there are more available tools on the toolbar now. The architecture provides wider choice of tolls to operate.



Now the required (white) interface is added. It is simultaneously assigned to the implemented frame. It has totally same properties as the provided one. The only difference is type. But in fact both required and provided interfaces are instances of the same class. They are used for different purposes.

The last way of adding interface is to add it on subcomponent's border. The creation of the subcomponent will be described in the next chapter. Now we will illustrate how to add an interface to a subcomponent.

We instantiated subcomponent by a frame without interfaces. So, the subcomponent is empty now.

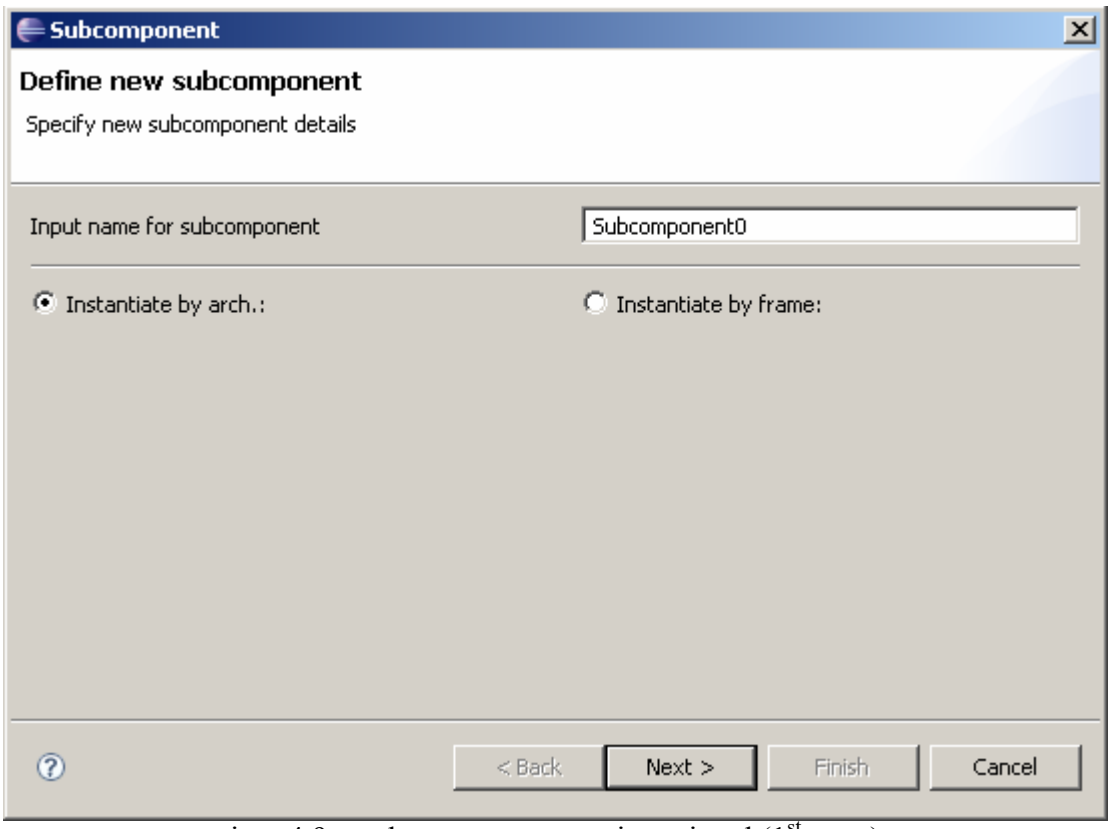


img 4.9a – interface on a subcomponent.

This was the last way of creating interfaces. Interfaces on a subcomponent are directly connected with instantiated entities. This means that as soon as a new interface is added, it is assigned to instantiated architecture and/or frame. But as was told before, it is possible to use the one interface instance on different entities for different connections.

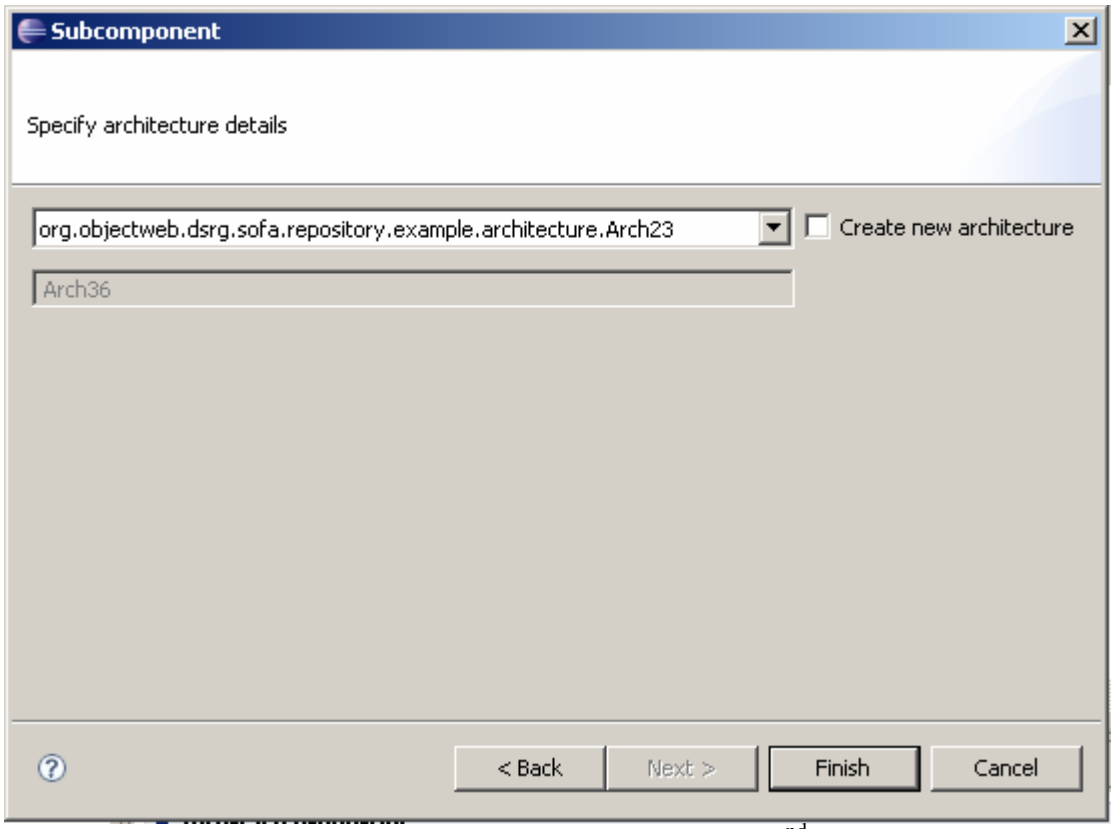
3.3.5 Create subcomponent

The only possibility to create a subcomponent is to create it on architecture. Subcomponent's creation does not depend on architecture or frame centric view. It is just required to have an architecture drawn on the draw area. Then we should have the 'Subcomponent' button active. Click on this button and the cursor is changed to the cross sign. Then click over any place above the architecture. This will bring subcomponent creation wizard.



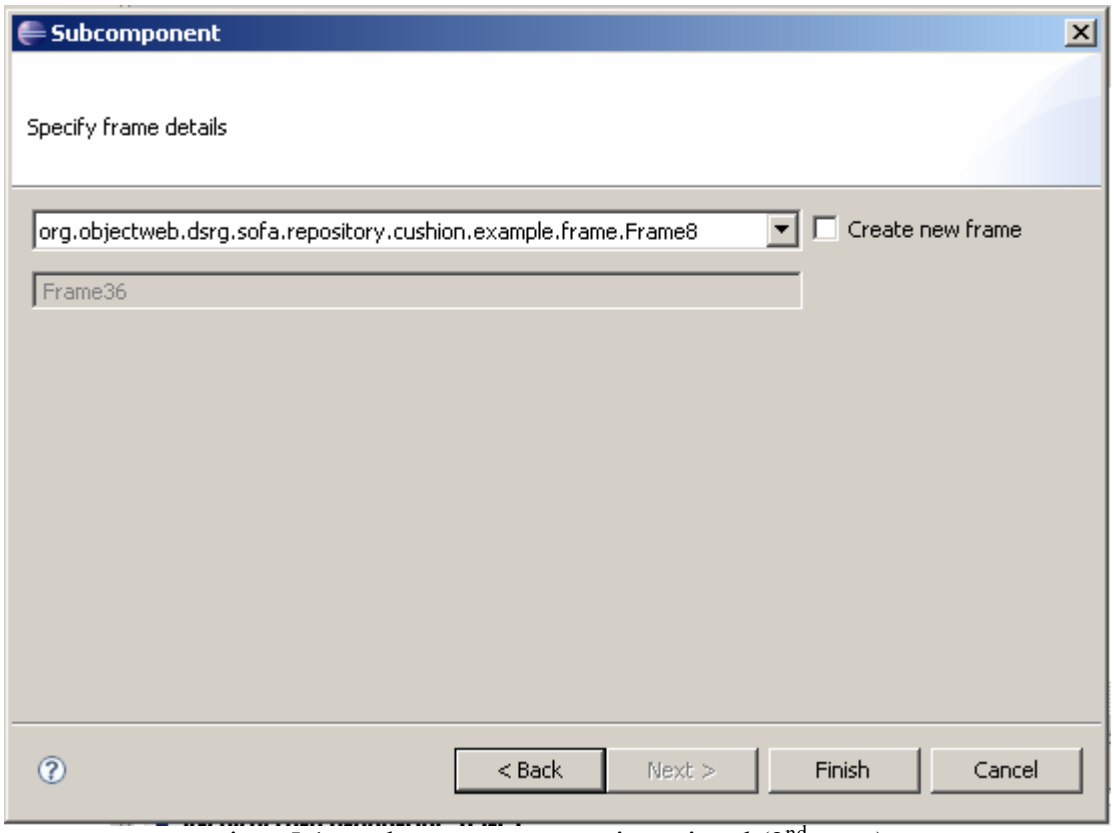
img 4.9 – subcomponent creation wizard (1st page)

The first page of the wizard asks to input name of the subcomponent and select type of entity (architecture or frame), which will instantiate the new subcomponent. When all data is set we can proceed to the next page. Now we instantiate subcomponent by architecture.



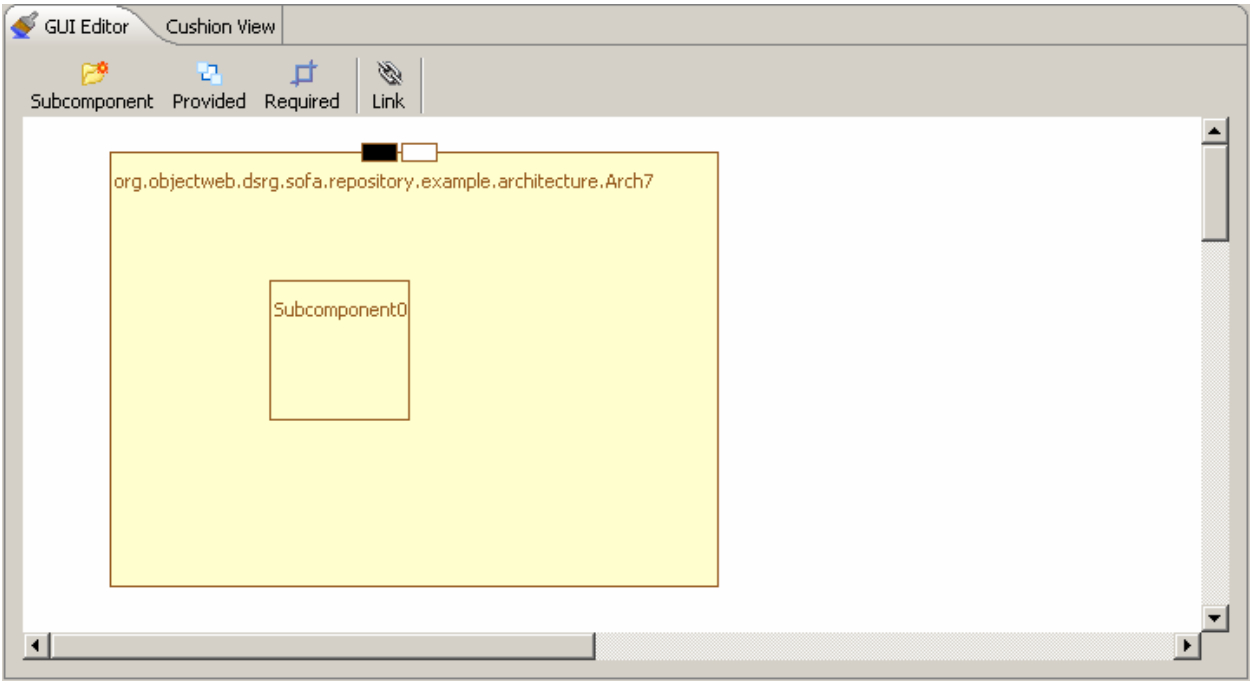
img 5.0 – subcomponent creation wizard (2nd page)

The second page offers to select an existing architecture from the drop down list or create a new one. It is recommended to use existing architectures for instantiation. Now let's come one step back and try to instantiate a subcomponent by a frame. To do this we should just switch radio button to the 'instantiate by frame' position.



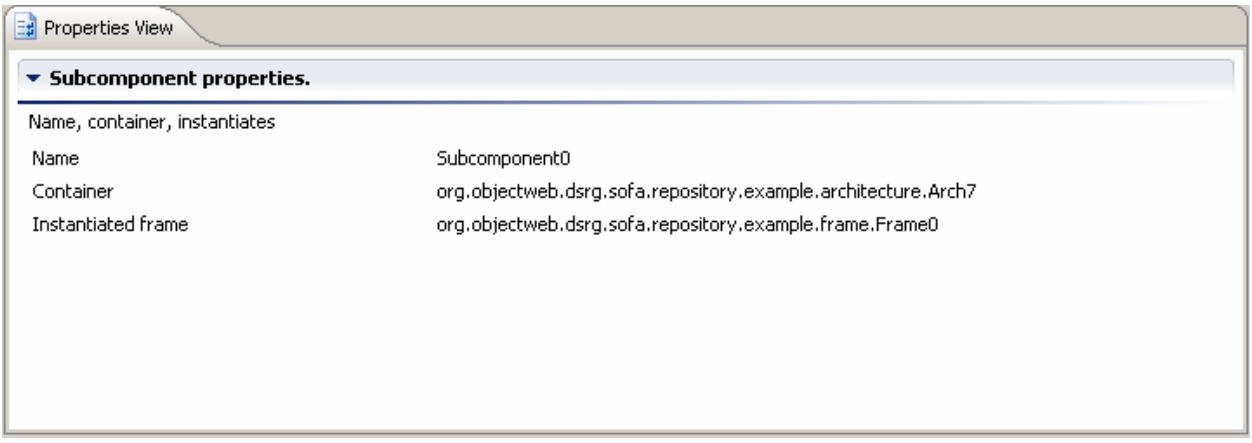
img 5.1 – subcomponent creation wizard (2nd page)

Here we can choose to create new frame or to select existing one. This is not really important. The drop down list is filtered and it is possible to select only frames which do not make any collision in existing repository. Now we click finish and the subcomponent is created.



img 5.2 – subcomponent is created

The subcomponent is created and allocated on the architecture. It has only 3 properties and all of them are read only. So this is not too much of interest. You should remember that subcomponent's name can not be changed after it is created. This is valid for frames, architectures, and interface types. All of them have read only name properties.

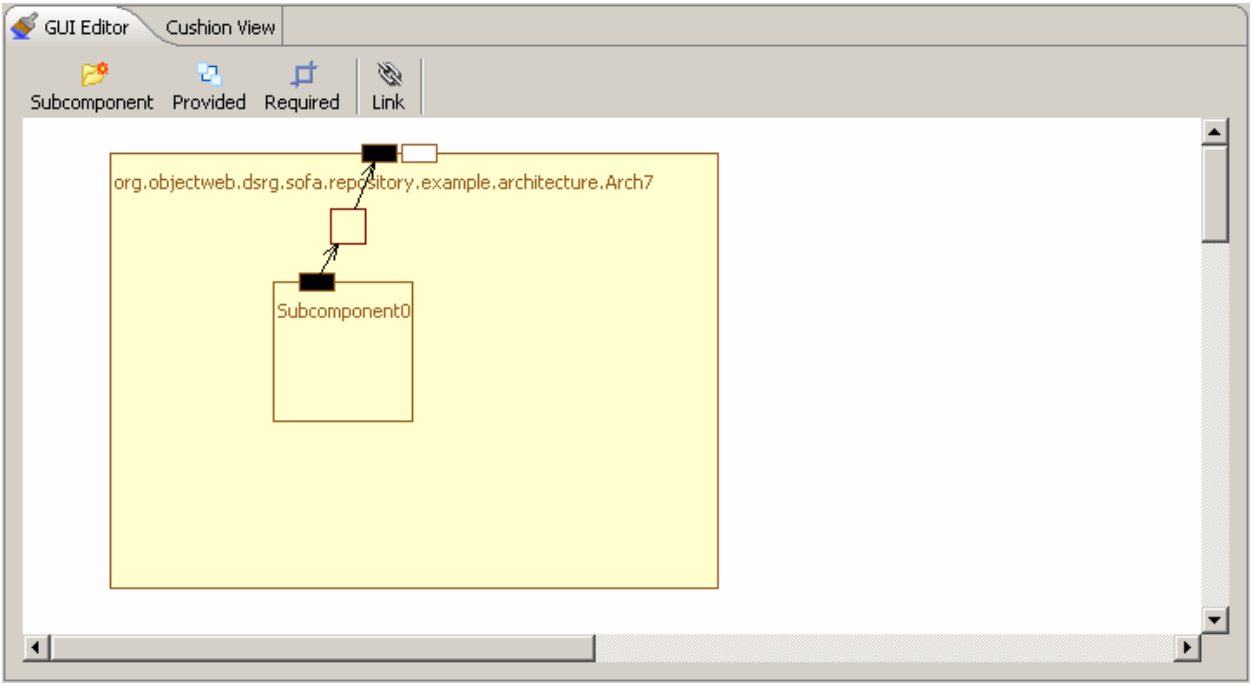


img 5.3 – subcomponent's properties

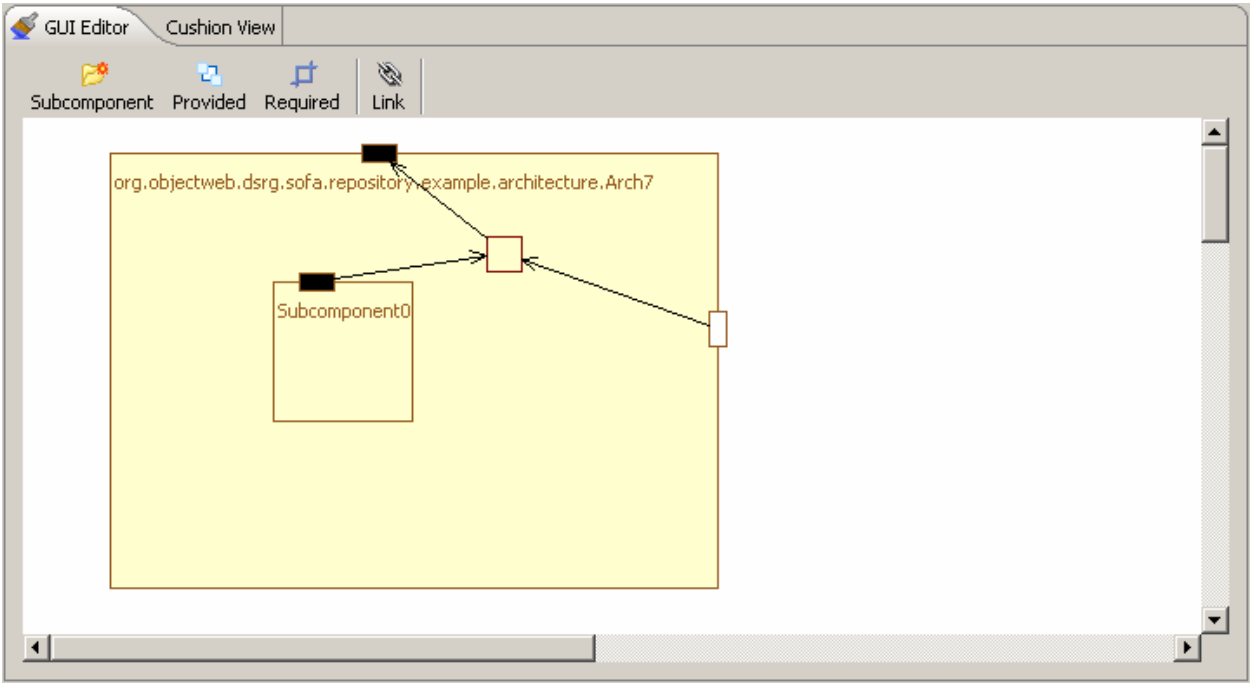
3.3.6 Links

The link is created in this way. It is required to click on the link button (

), which is located near the required interface button on the GUI editor toolbar. Then click on an interface and drag mouse (do not release button after the click on the first interface is done) to another interface. Release mouse's button when the mouse cursor is over the other interface. This should create a link with a square in the middle. Square means that link is multi-connectional. This means than many interfaces can be connected by one link. To add an interface to the connection you should click the link button, then click on the interface which should be added and do not release clicked mouse button. Then drag the mouse over link's square. When the cursor is over the square – release the button.



img 5.4 – new link is added.

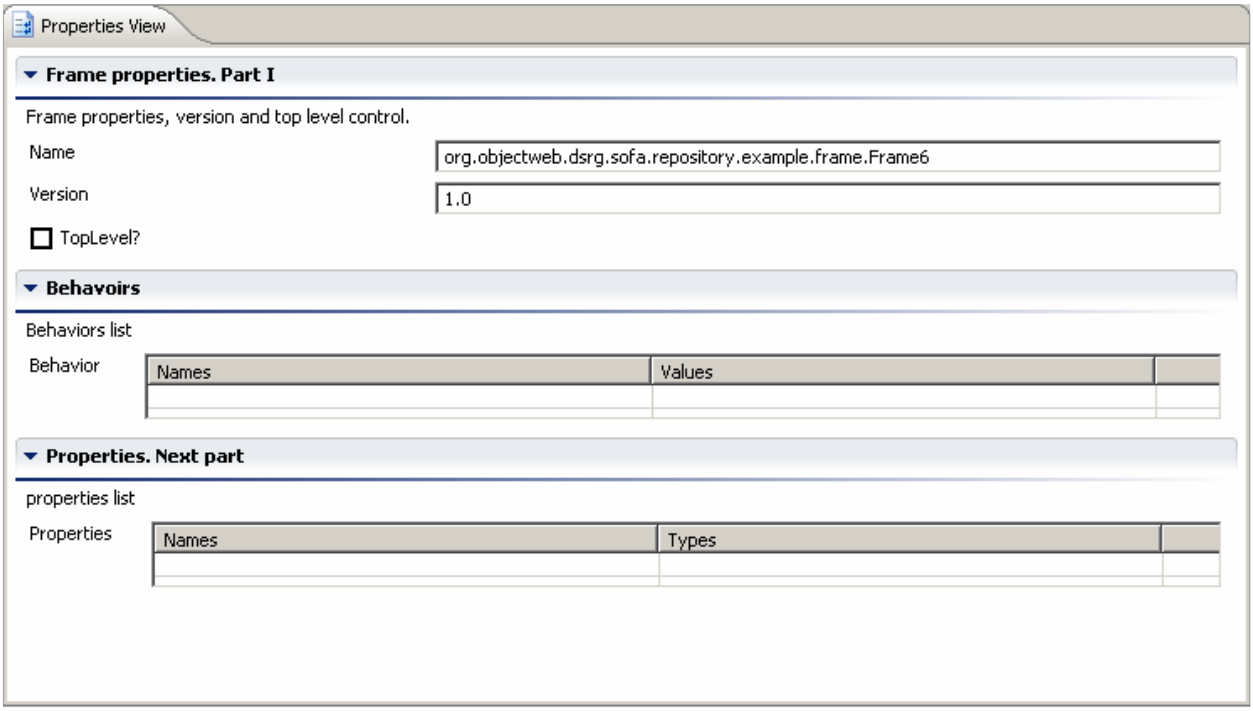


img 5.5 – new interface is connected

Link does not have properties. When a link is selected, only its type is shown. And all links in the repository are connectors.

3.3.7 Frame properties

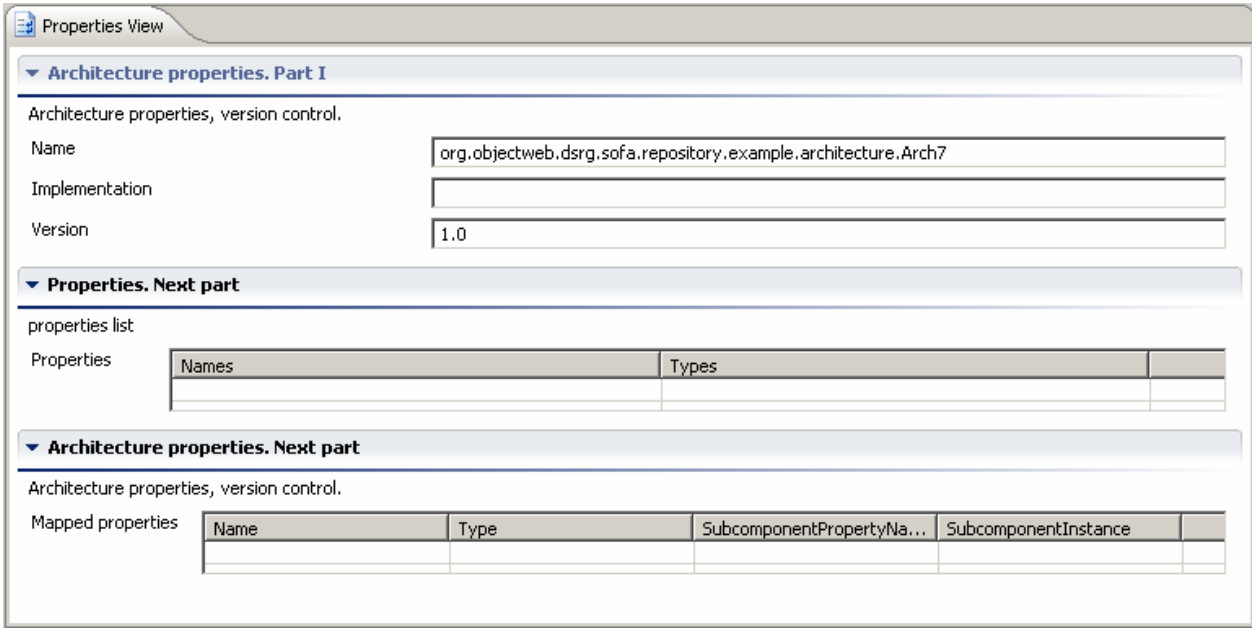
Each frame has properties. This includes name, which is read only. The name can not be changed after the frame is created. Then it includes version. The version is important property. When version is changed, the entity is updated as versioned in the repository. Each version of an entity is stored in the repository. So, it is possible to checkout repository using cushion tool and receive needed version. Also frame has top level property. This is Boolean flag . When it is raised, the frame is declared as a top level frame.



img 5.6 – frame properties.

Then there is list of behaviors. This list is introduced instead of one protocol property. Now it is possible to set many protocols for one frame. To add a behavior to the list it is necessary to click on the table. To remove a behavior it is enough to select a behavior (or many behaviors) by a mouse click and press the 'Del' key. The properties list works in the same way the behavior list works. The only difference – it has two columns.

3.3.8 Architecture properties



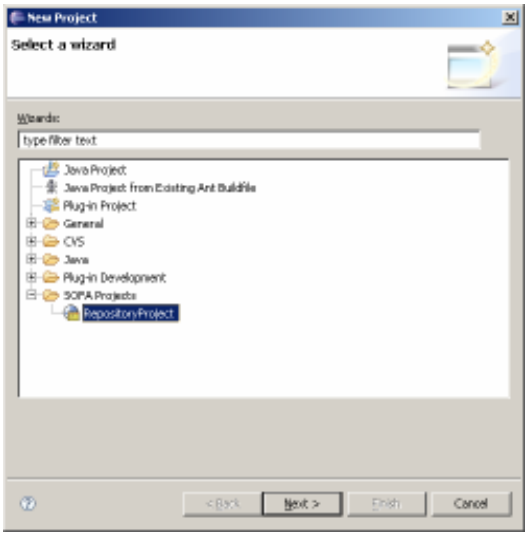
img 5.7 – architecture properties.

3.4. Working with cushion extension

Cushion extension allows local generation and further editing of the SOFA 2.0 entities.

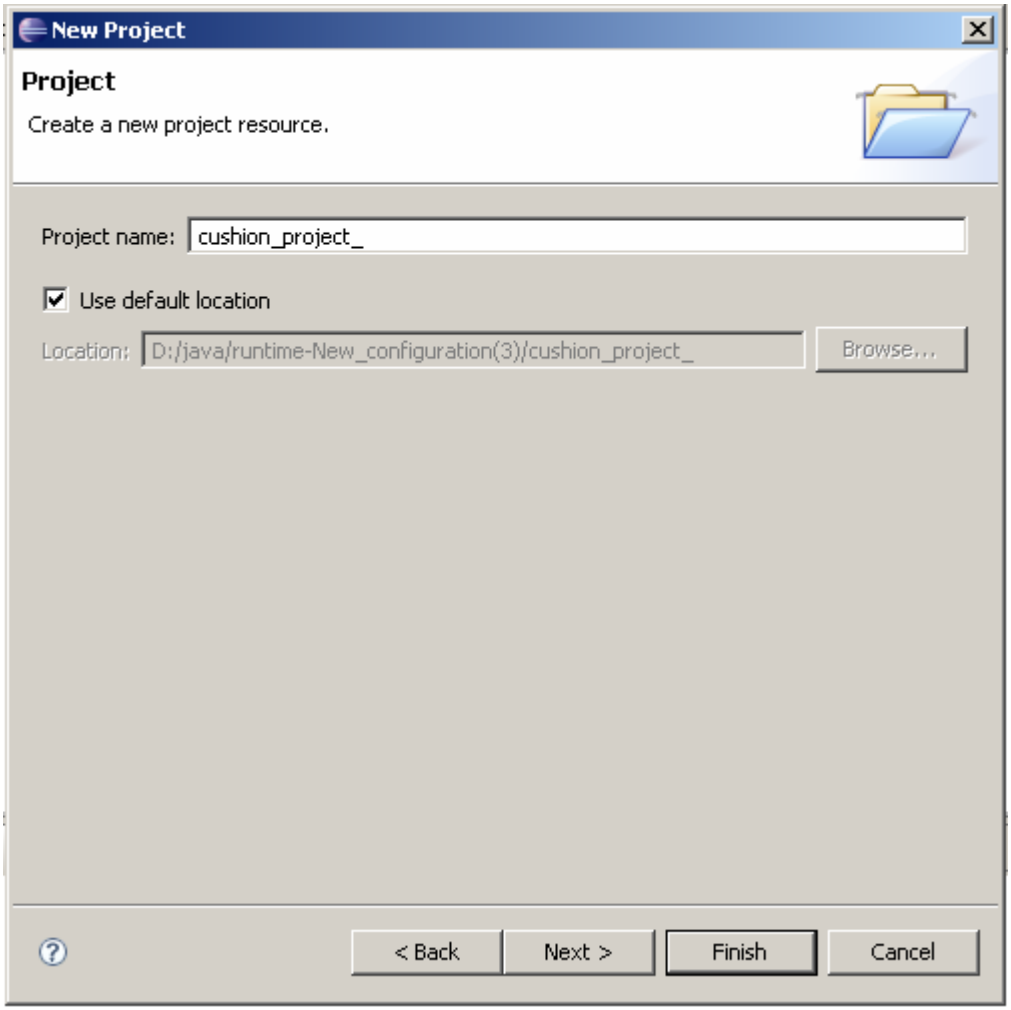
3.4.1. Creating new cushion project

To create new cushion project it is required to open the File->New->Project menu or click the button 'New' and select the Project from the drop down list of options.



img 5.8 – create new cushion project

Then select SOFA 2.0 projects -> new repository project. In the wizard it is required to type any desired name of a project to be created. Then the folder with typed name will appear in the Package Explorer.



img 5.8 – create new cushion project (wizard page)

The new project will be created in the current project workspace. It does not differ from other project created in the Eclipse IDE until the first cushion entity is generated. Cushion generates

file _config.cfg which is used to track all versions and names of entities generated within the current project. It generates all entities in this way: when a creation wizard is finished, and all data from user is collected, it creates a folder named as new entity should be called and adl.xml file inside this folder. When user wants to edit properties of an entity, it is required to edit this adl.xml file inside the created folder. There is special multi-page editor for these purposes.

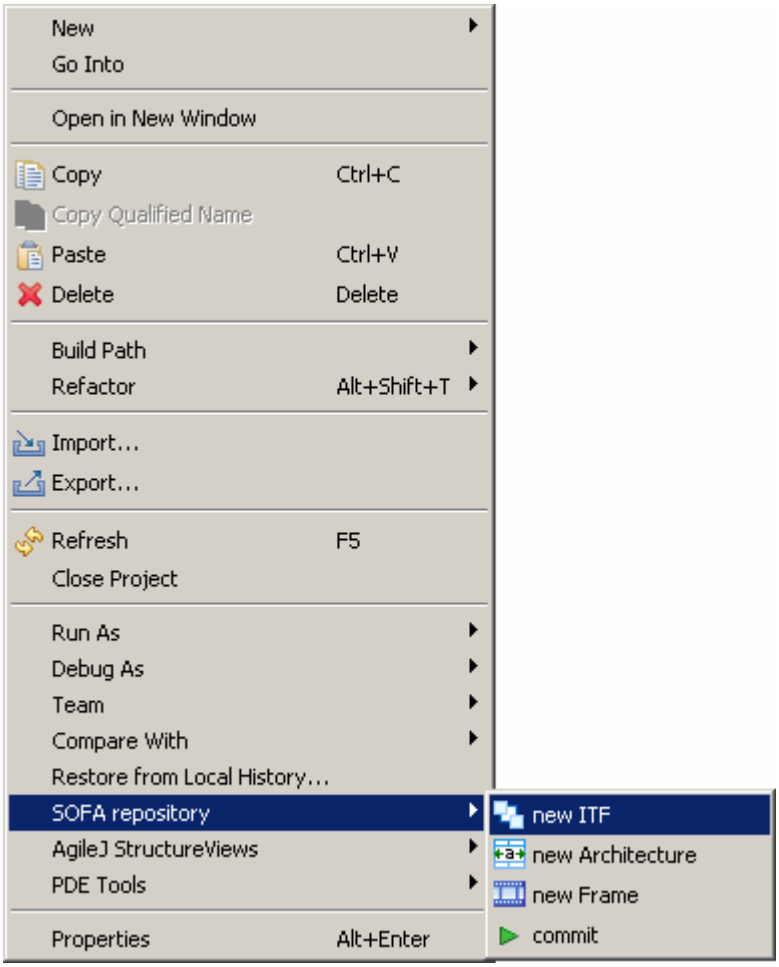
3.4.2. Package explorer, cushion view, multi-page editor

All cushion projects are visible from the Package Explorer, the default Eclipse package explorer. To edit properties of any entity it is required to open it in the editor from the Package Explorer. New entities can be generated using contribution to the context menu of the Package Explorer. Cushion view is used to display graphical representation of a currently edited entity. It looks like almost the same as the GUI editor for the direct repository editing, but the main difference is the cushion view does not have tool bar with buttons. All interfaces and subcomponents are created using appropriate buttons on the multi-page editor's pane. The multi-page editor itself is the editor part which is opened when any adl.xml file is double clicked from the Package Explorer. It is impossible to avoid its opening and substitute editor with a view. This is due to closed architecture of the Package explorer.

3.4.3 Create new interface type

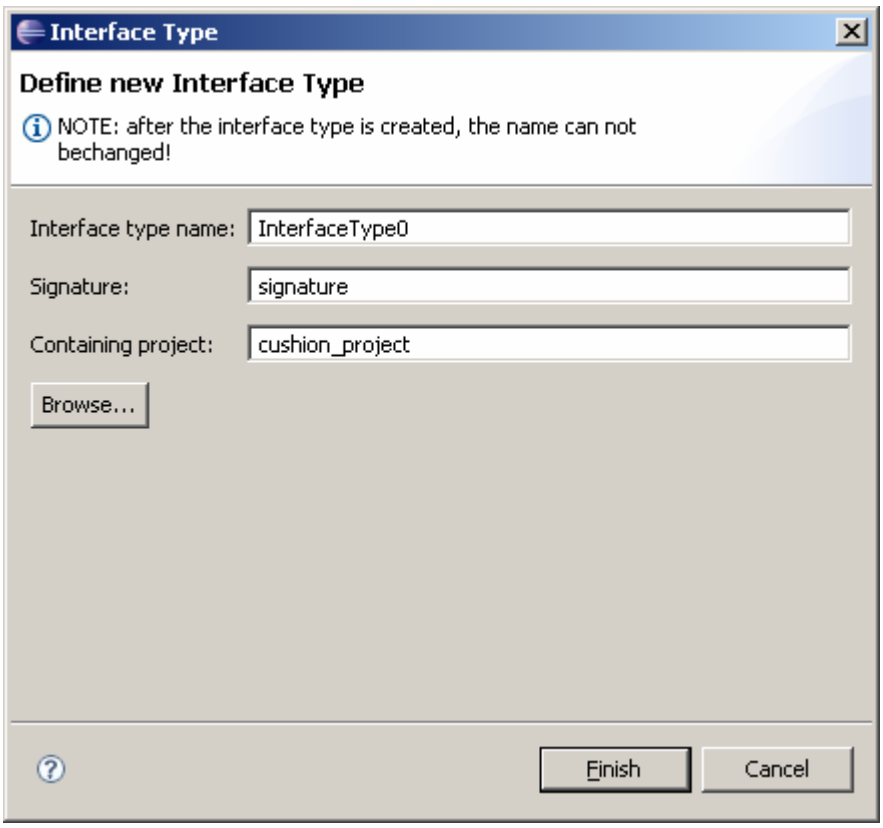
The first step in developing the project is to create interface type. It is possible to create any cushion entity at any time, but logically it is better to create interface types first, then frames, and only then architectures. This is because entities should be committed in this order. Frames can not be committed until interface types which are used by interfaces are committed, and architectures can not be committed until implemented frames are committed. That is why we recommend this order of entities creation.

There are 2 ways to create each cushion entity. The first we can call appropriate creation wizard from the context menu of the Package explorer, and the second way is to use File->New->Other->SOFA projects->New… menu. The interface type is not exception in this case.



img 5.9 – context menu to create new interface type

To create new interface type from the context menu right mouse click on the project's name or any file inside the project. Then select SOFA repository->new ITF option. This will bring the same wizard as for the File->New->Other->SOFA projects->New ITF option.
The wizard asks to input all interface type properties: name and signature. When this is done, it is not necessary to edit the interface type at all (if you are not going to change signature for sure). After the interface type is created, it can be committed at once.

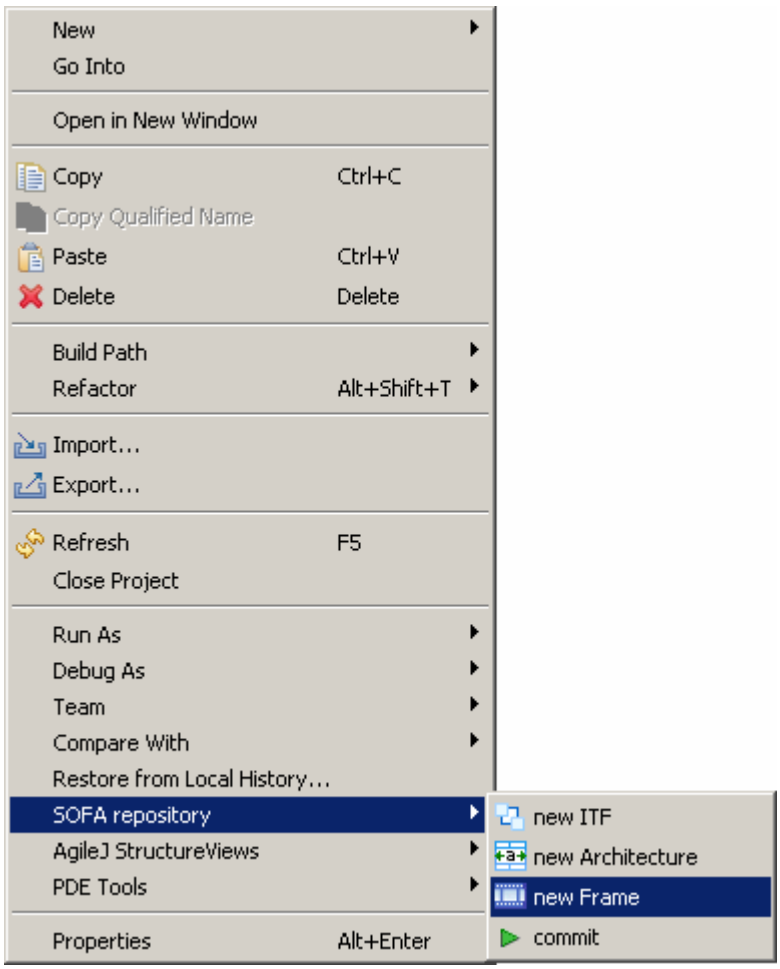


img 6.0 – new interface type wizard.

It is also possible to select containing project for the interface type. By default this is currently open project. But if user has more than one project opened, it is possible to select another project using the Browse… button, or just type project's name in the appropriate text field.

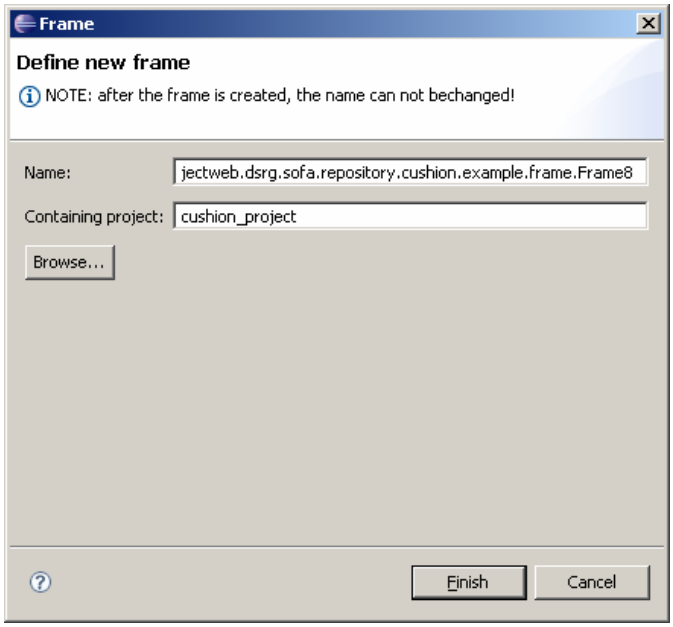
3.4.4 Create new frame

After all needed interface types are created (and possibly committed to the repository server) we can proceed to the next step and start to create frames. In fact there is no difference between interface type and frame in calling the appropriate creation wizard from the context menu or from the File->Other->SOFA project->New frame menu. Except user shell select all menus as for the new frame instance.



img 6.1 – context menu to create new frame

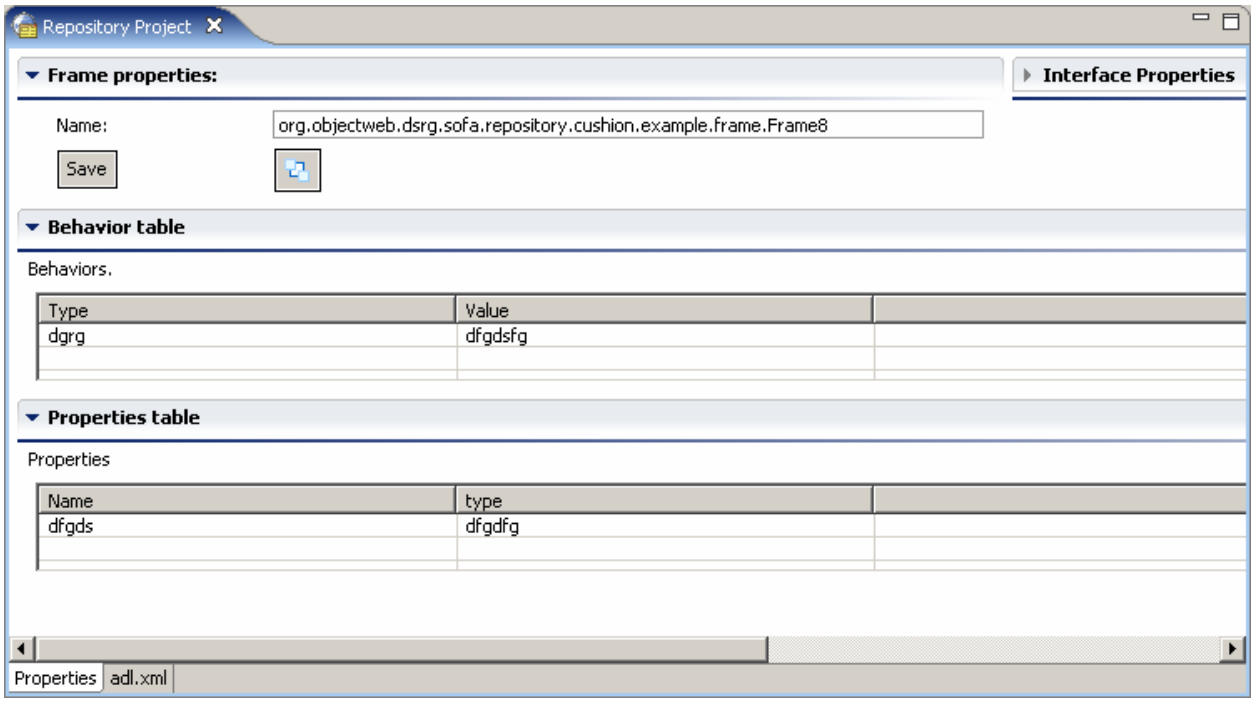
To create new frame from the context menu right mouse click on the project's name or any file inside the project. Then select SOFA repository->new Frame option. This will bring the same wizard as for the File->New->Other->SOFA projects->New Frame option.
The wizard asks only for name of the newly created frame. All other properties should be edited via the multi-page properties editor after the frame is created.



img 6.2 – new frame wizard

It is also possible to select containing project for the frame. By default this is currently open project. But if user has more than one project opened, it is possible to select another project using the Browse… button, or just type project's name in the appropriate text field.

After the frame is created, it is visible in the list of entities in the package explorer. To edit its properties, expand the frame's folder and double click (or hit the enter key) on the adl.xml file. The multi-page editor should be opened.



img 6.3 – frame properties.

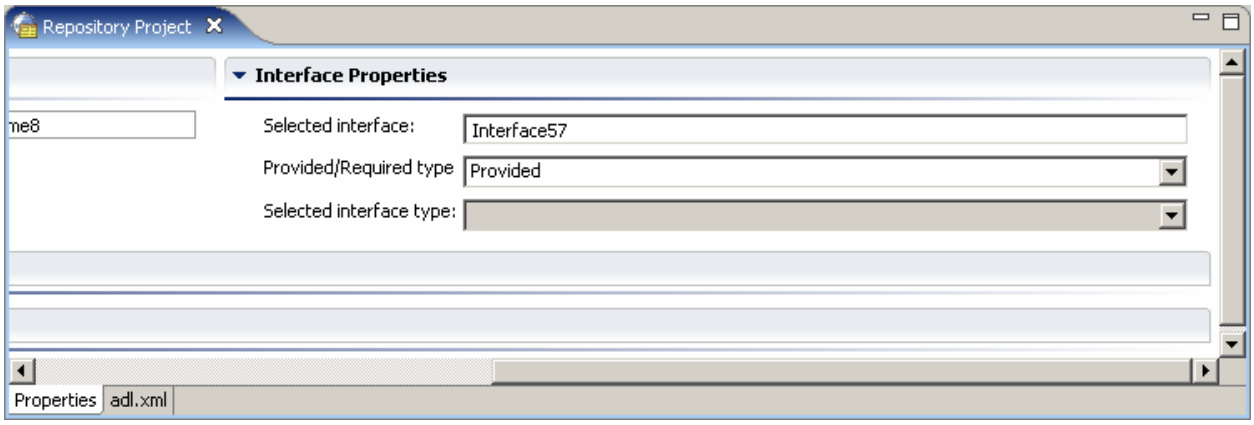
The frame properties include frame name (which can not be edited), list of behaviors and list of properties. Both lists can be easily edited. To delete one or more rows from tables select row using mouse (and ctrl/shift key to select multiple) and hit the 'Del' button on the keyboard.
In the properties list you can also see button to create new interface and the interface properties tab which is not active until any interface is selected.

3.4.5 Create new interface

To create an interface it is required to have an editor for frame properties opened. When frame properties editor is open, then it is needed to click the 'new interface' button and new interface will be automatically added on the border of the currently active frame. By default it is provided (black), it is possible to change its type in the interface properties when the interface is selected.

img 6.4 – new interface on frame's border

The interface should be selected (its borders should be green) in order for the Interface properties tab to be enabled in the Frame properties editor.

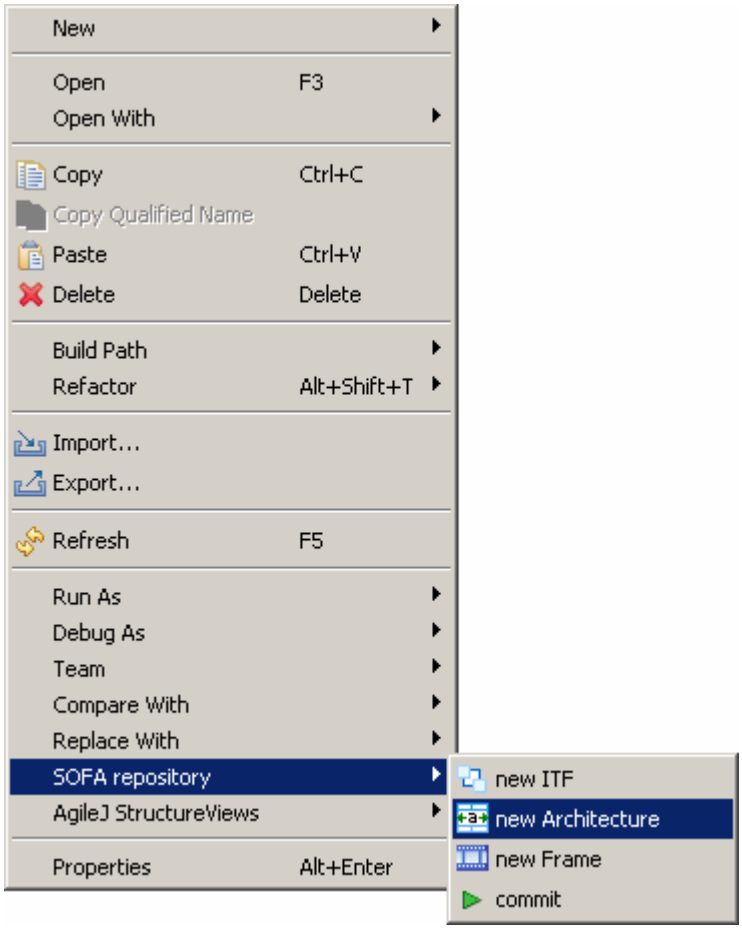


img 6.5 – interface properties tab

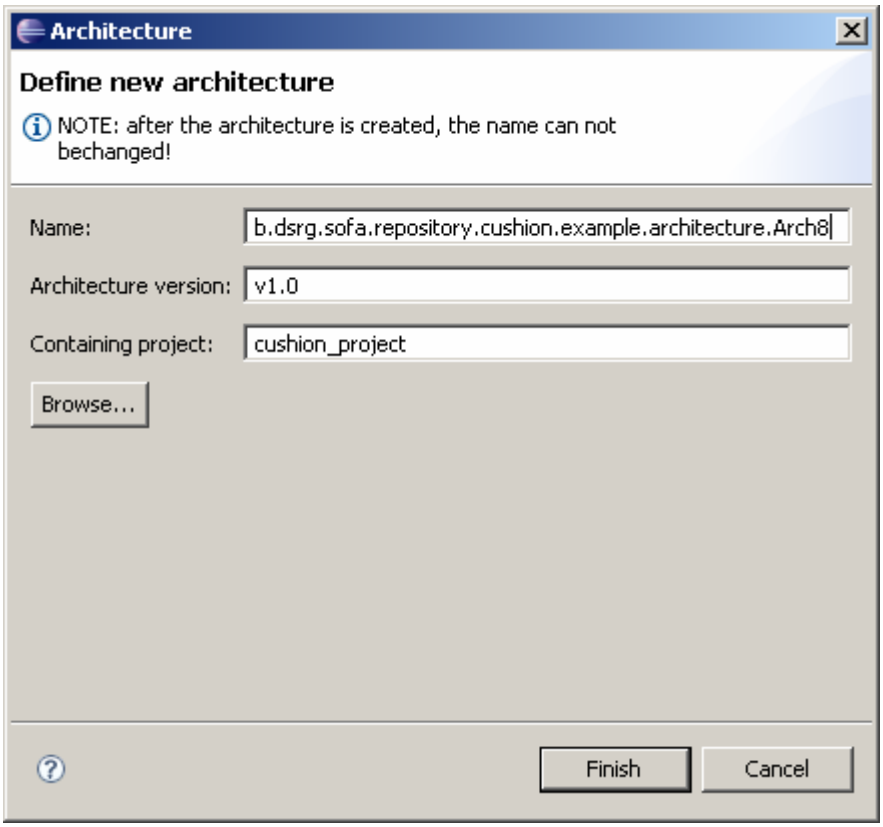
The interface properties tab allows to change name of the interface, change its type (provided/required) and set interface type which will be assigned to the interface. It is also possible to delete an interface when it is selected just by hitting the 'Del' key on the keyboard.

3.4.6 Create new architecture

After all needed frames are created (and possibly committed to the repository server) we can proceed to the next step and start to create architectures. In fact there is no difference between architecture and frame in calling the appropriate creation wizard from the context menu or from the File->Other->SOFA project->New architecture menu. Except user shell select all menus as for the new architecture instance.



To create new architecture from the context menu right mouse click on the project's name or any file inside the project. Then select SOFA repository->new Architecture option. This will bring the same wizard as for the File->New->Other->SOFA projects->New Architecture option.
The wizard asks only for name of the newly created architecture. All other properties should be edited via the multi-page properties editor after the frame is created.

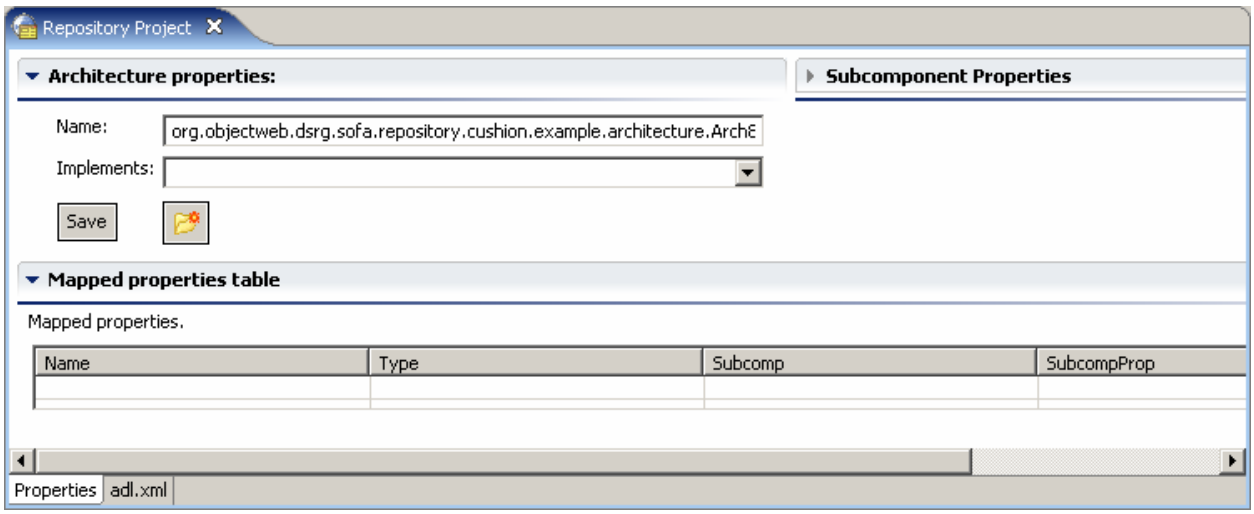


img 6.7 – new architecture wizard

The wizard also offers to input version value, but it is really optional. The version will be generated automatically later, when the engine will generate adl.xml file and store changes to the _cushion.cfg file.

It is also possible to select containing project for the architecture. By default this is currently opened project. But if user has more than one project opened, it is possible to select another project using the Browse… button, or just type project's name in the appropriate text field.

After the architecture is created, it is visible in the list of entities in the package explorer. To edit its properties, expand the frame's folder and double click (or hit the enter key) on the adl.xml file. The multi-page editor should be opened.

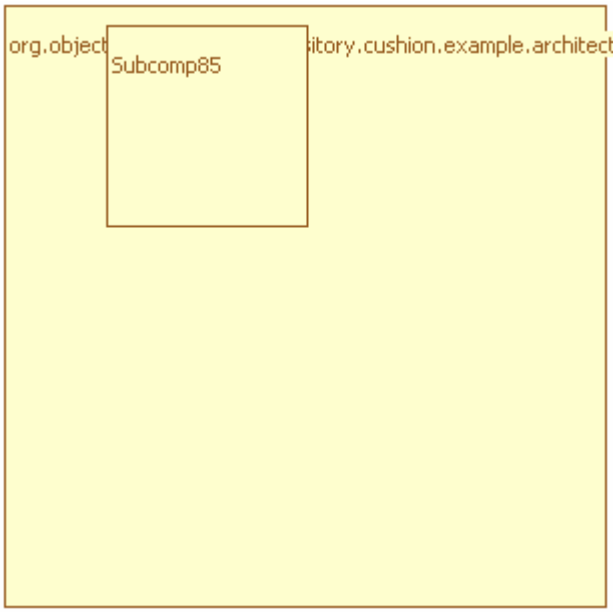


img 6.8 – architecture properties page

The architecture properties page allows to change mapped properties and set the frame to be implemented by the architecture. The frame must be set in order to commit architecture properly. Mapped properties are optional. The properties editor also provides the 'New subcomponent' button and the subcomponent properties tab. This tab is active only when a subcomponent is selected.

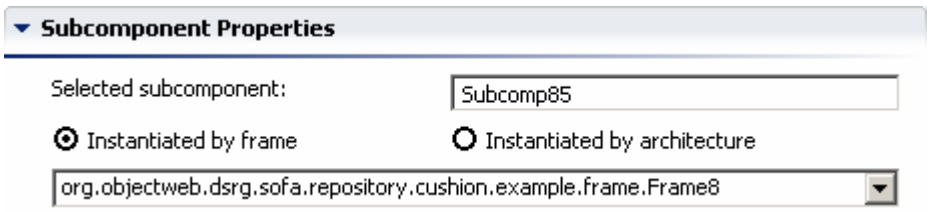
3.4.7 Create new subcomponent

In order to create a new subcomponent, it is required to have an architecture opened for editing. When the architecture is open, then click on the 'New subcomponent' button near the save button and new subcomponent will be automatically placed on the architecture.



img 6.9 – subcomponent on the architecture

The subcomponent should be selected (its borders should be green) in order for the Subcomponent properties tab to be enabled on the Architecture properties editor. The selected subcomponent can be easily deleted just by hitting the 'Del' key on the keyboard.

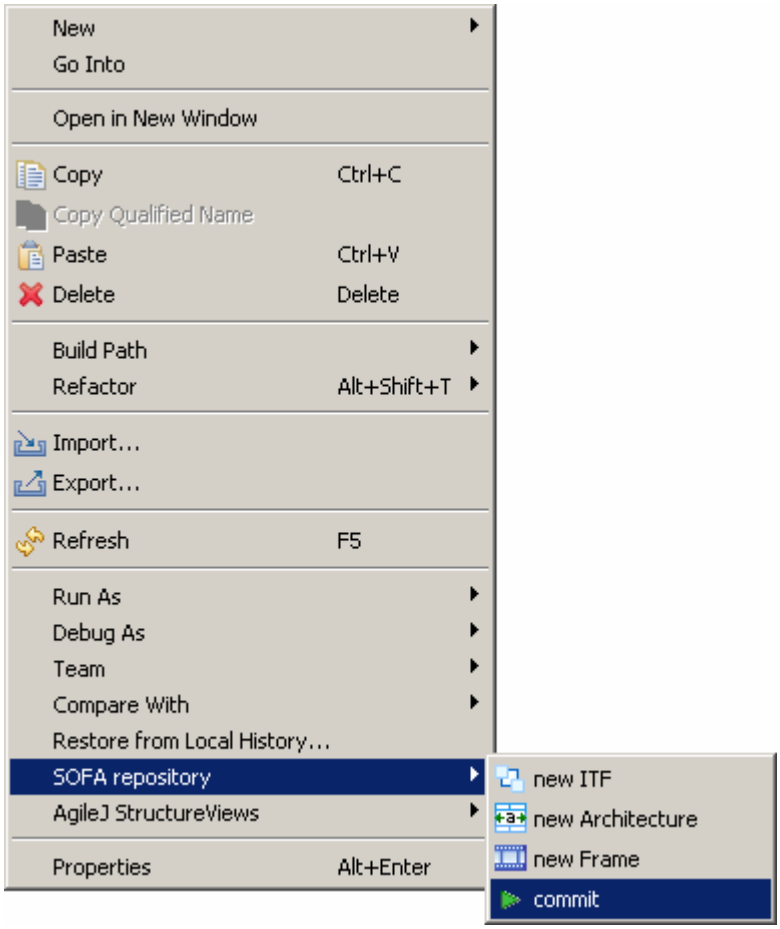


img 7.0 – subcomponent properties tab

The subcomponent properties tab allows changing the name of the subcomponent, and setting of the instantiated entity (architecture or frame). The instantiated entity must be set in order to commit architecture properly. As soon as user switches between possible frame and architecture alternatives, the drop down list is refreshed accordingly.

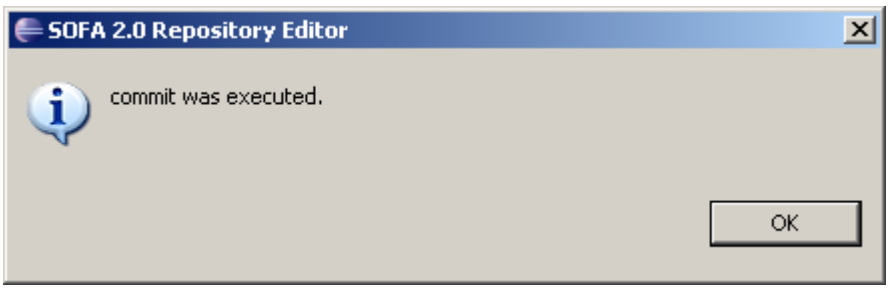
3.4.8 Committing to the repository

When all work is done, it is useful to save it and then commit all changes to the repository. Saving can be done by the save button directly on the property editor pane or by the global save button on the Eclipse toolbar – this is regular save button which is used to save changes to any file changed in the Eclipse IDE, do not mix with save all changes to the repository button.



img 7.0 – committing changes

The commit process can be performed in this way: select an entity (frame, architecture, or interface type) from the Package Explorer preview and then call the context menu by the right mouse click. Then select commit from the SOFA repository menu. This will commit only selected feature. If the commit is successful, user should receive the successful commit message. And an error can be thrown otherwise.

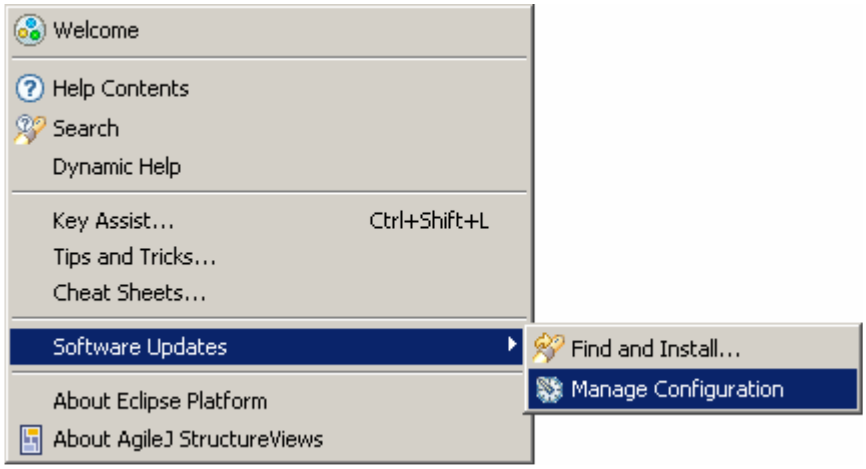


img 7.1 – successful commit

But it is preferable to commit all changes at once – select the root folder (name of the project) and perform all the same things as for the 1 item commit. This means right click on the root folder of the project, then select commit from the SOFA repository menu and wait until a message is shown.

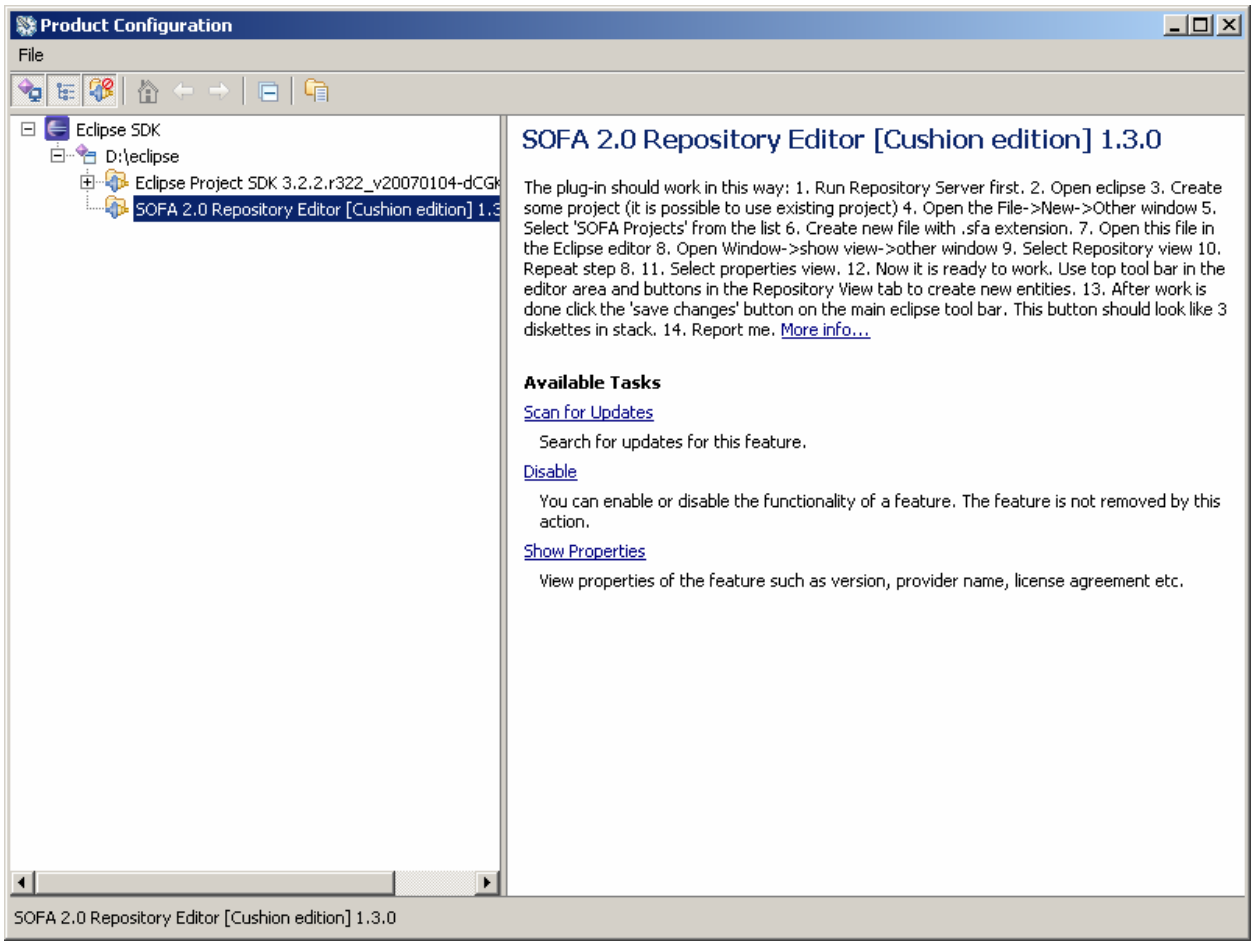
Chapter 4. Uninstall the plug-in

The Eclipse™ IDE provides the uninstall feature for all properly installed plug-ins. This feature is mainly available for those plug-ins which were installed via the local or remote site option. The uninstall process can be performed in this way: select Help -> Software Updates -> Manage Configuration wizard.



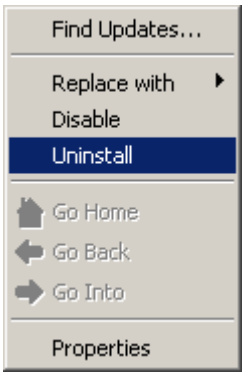
img 7.2 – Configuration manager menu path

The opened wizard is the configuration manager. It allows to manage existing plug-ins. Using configuration manager user can scan for updates for a plug-in, disable plug-in or view plug-in's properties. The uninstall feature of the configuration manager is accessible from the right click context menu in order to avoid unwanted or occasional uninstalls.



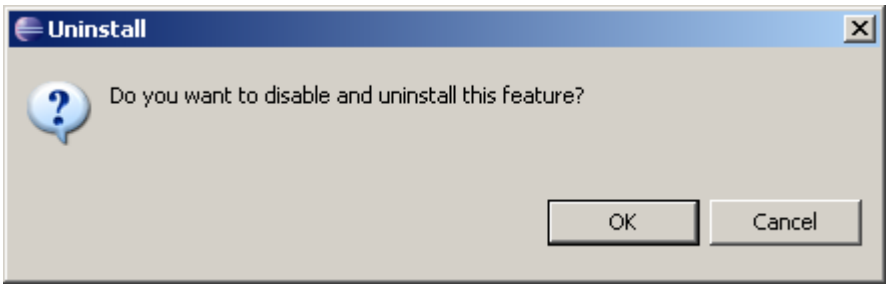
img 7.3 – Configuration wizard

To completely uninstall the plug-in user should select the plug-in in the tree view of plug-ins and right click on it. Then select the uninstall option from the shown menu.



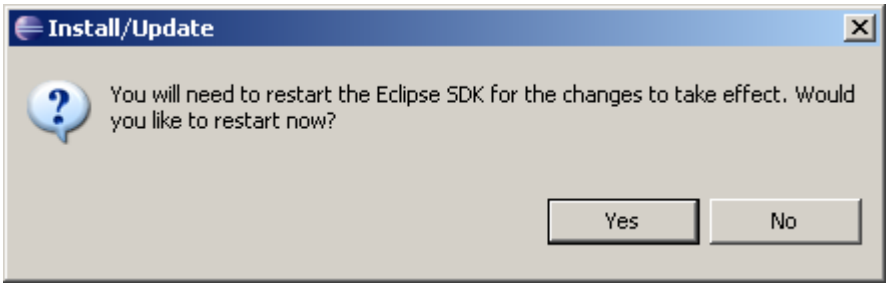
img 7.4 – Uninstall popup menu

The final warning will be thrown and if user confirms that it is really required to disable and uninstall the plug-in, it is uninstalled and unplugged from the workbench.



img 7.5 – Uninstall confirmation

After the uninstall is done and the plug-in does not exist in the workbench, the Eclipse IDE should be restarted to clean the memory from unwanted references. Just confirm the restart request.



img 7.6 – Restart confirmation

Chapter 5. Support, troubleshooting, further information

The plug-in is currently maintained by Maksym Nesen. All support requests, bug report, issues, proposals etc should be forwarded to him directly. The contact e-mail is nesen@dsrg.mff.cuni.cz or mavines@oksima.biz. All requests will be carefully solved and all questions will be answered.

Chapter 6. Open issues

The plug-in still has some open issues. This includes implementing full cushion support, code bundle support, connections list, and possibility to choose connection to open/close, and some more.